Lotus

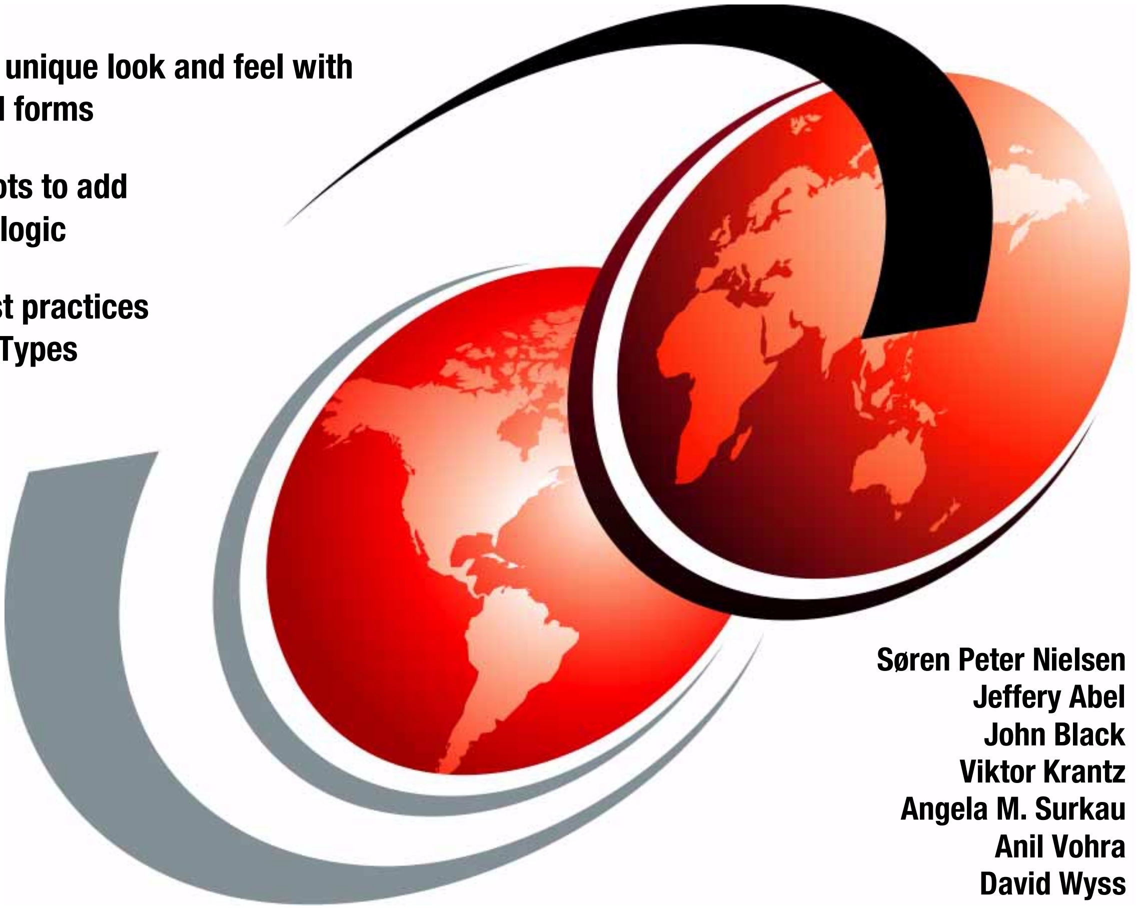
IBM

Customizing QuickPlace

Create your unique look and feel with Themes and forms

Use PlaceBots to add application logic

Capture best practices using PlaceTypes



Søren Peter Nielsen
Jeffery Abel
John Black
Viktor Krantz
Angela M. Surkau
Anil Vohra
David Wyss

Redbooks

ibm.com/redbooks

SG24-6000-00

International Technical Support Organization

Customizing QuickPlace

December 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix M, “Special notices” on page 427.

First Edition (December 2000)

This edition applies to Lotus QuickPlace Release 2.0 and 2.0.5

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. TQH 1CP-5605E
2455 South Road
Poughkeepsie, New York 12601-5400

Contents

Preface

Lotus QuickPlace is the leading self-service Web tool for team collaboration. This IBM Redbook details how you, as a Place owner, a Web designer, a programmer, or a QuickPlace administrator, can take QuickPlace to the next level through customization.

We show you how to apply your own unique graphic design to the QuickPlace user interface, and how to add functionality using JavaScript. We show how you can develop Forms (built-in, MS Office, or HTML forms), and use the built-in workflow to support your process. Then we show how you can add automation by programming PlaceBots (agents) in LotusScript or Java. To develop advanced customizations in QuickPlace, you have to understand the internals and how to modify them. We discuss this in chapters on the QuickPlace Object Model and the QuickPlace Developer's Kit.

Even though QuickPlace is a self-service tool, there will often be a need to integrate with other systems. Therefore, we show how you can integrate with Notes, Domino, Domino.Doc document management, and the Domino Workflow product. We also discuss how to integrate QuickPlace in a portal.

Finally, we show how you can clone the Places you choose, together with their customization as PlaceTypes, and how to build a Turnkey server with your PlaceTypes and other customizations, for internal reuse or for resale.

Many examples are included throughout the book.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Cambridge Center.

Søren Peter Nielsen works for the International Technical Support Organization at Lotus Development, Cambridge, Massachusetts. He manages projects that produce redbooks about all areas of Lotus products. Before joining the ITSO in 1998, Søren worked as an IT Architect for IBM Global Services in Denmark, designing solutions for a wide range of industries. Søren is a Certified Lotus Professional at the Principal level in Application Development and System Administration.

Jeffery Abel is a Senior Systems Engineer for Lotus Development in Richmond, VA, USA. He specializes in information systems, IT strategic visioning, and the development of IT solutions for customers using Lotus technology. Jeffery has also managed application development for a large power company, building Domino Web applications and integrating Domino with other business applications and ERP systems.

John Black designs Knowledge Management solutions for the United States Department of Defense. While producing a visionary enterprise information-sharing portal, he has sponsored a QuickPlace beta test and a Raven test build program. He is a practitioner of Goal-Directed Design®, a process uniting conceptual design, interaction design, and interface design. He is experienced in Lotus application development and business process re-engineering.

Viktor Krantz is Senior Web Developer and e-Business Consultant at Strategic Net Applications, Inc., an IBM and Lotus Business Partner in Overland Park, Kansas. He works with an e-Business development team, developing solutions that range from client Web sites to e-commerce and Intranets that integrate back-end systems. He has extensive experience with graphic and Web development software. He also has 12 years of teaching experience on a number of software platforms. He has worked with Lotus Notes since Version 3, and has been focusing on Domino Web application development since Version 4.0. His e-mail address is vkrantz@snapps.com.

Angela M. Surkau is a Consultant and Project Manager for Web Application Development at FoxCom OHG, an IBM and Lotus Business Partner in Nauheim, Germany. She works with an e-business team developing Web applications, ranging from small Web sites to Web applications that integrate with Domino and back-end systems on the AS/400. Angela also has several years of experience administrating NT and AS/400. Her e-mail address is asurkau@foxcom.de.

Anil Vohra is a Consultant Specialist with e.Messaging at EDS in Troy, Michigan, USA. He has over 15 years of experience in designing distributed systems and messaging solutions for a wide range of industries. Anil is a Microsoft Certified Systems Engineer and a Certified Lotus Professional at the Principal level in Application Development and System Administration. His e-mail address is anil.vohra@eds.com.

David Wyss is a Domino developer for Lotus Switzerland. David specializes in the development of Web applications based on Domino, using primarily LotusScript, JavaScript, DHTML and Java. David is currently developing browser-based business applications for a number of major European corporations, as well as making Web technology presentations at an international level.

A number of people have provided support and guidance. Special thanks to **Albert Whitehead** from Whitehead and Associates Inc. for writing the appendix about special AS/400 considerations, and **Justine Middleton** from ITSO Rochester for managing the content creation from Albert Whitehead and Anil Vohra.

Special thanks also to **Liz McKay**, Product Manager, **Brennan O'Hara**, Marketing Manager, **Julio Estrada**, Development Manager and the whole QuickPlace development team.

In addition, we would like to thank the following people:

- Anna Barr, Logicon, Inc.
- Victor Brown, Logicon, Inc.
- Steve Cappo, Lotus
- Ian Connor, Iris
- Marianne Goggin, domEkno.com
- Charlie Hill, Iris
- Axel Lachmann, FoxCom OHG
- Rob Novak, Strategic Net Applications, Inc.
- Jim Ott, Lotus
- Greg Pickrel, Lotus
- Erin Reily, Lotus
- Sami Shalabi, Iris
- Patrick Shaughnessy, Iris
- Alan Thomas Payne, Internet AG
- Cees Van Der Woude, Lotus
- David Morrison, ITSO
- Terry Barthel, ITSO
- The ITSO Poughkeepsie editing team

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 437 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Introduction

Lotus has long been an innovator in technology that enables people to work together. Lotus QuickPlace provides a revolutionary way to create collaborative spaces that exploit the Internet or your corporate intranet. Overcome time and distance, improve efficiency and share knowledge with this instant, affordable teamware from the world's leader in collaborative software solutions.

QuickPlace is truly an out-of-the-box usable product, but you can get even better solutions through customization of QuickPlace. This IBM Redbook is about customizing QuickPlace. It is targeted to QuickPlace developers, solution architects and, to some degree, administrators.

The structure of the book is the following:

- What is QuickPlace
- How to customize and why
- Installation
- Creating Themes and using the Graphics server
- Use of Forms
- Use of PlaceBots
- The QuickPlace Object Model
- The QuickPlace Developer's Kit
- Integrating with other applications
- Creating PlaceTypes and Turnkey Servers

In this chapter we give a brief introduction to QuickPlace and a list of terms you will encounter throughout this book.

1.1 What is Lotus QuickPlace?

QuickPlace is a self-service Web tool for *team collaboration*. QuickPlace enables the creation of a team workspace on the Web—instantly. Teams use QuickPlace to share and organize ideas, content and tasks around any project or ad hoc initiative.

QuickPlace is a *Teamware* solution. As an emerging category of software, International Data Corp (IDC) defines the attributes of Teamware as follows:

- Primarily designed for smaller groups or teams
- Duration of usage is often limited or for a specific period of time
- Easy to install, set up and get started
- Requires minimal infrastructure and support from IT
- Easy for users to create, manage and customize
- Project-centered or activity-oriented
- Strategic tool at the platform level; tactical tool at the team level
- Designed to facilitate enhanced collaboration
- Emphasis on user productivity and getting the job done

All of these Teamware attributes can be applied to QuickPlace. IDC predicts a worldwide market opportunity of $400M (USD) for Teamware solutions by the end of 2002, so there is ample opportunity to supply customized solutions for this market.

1.1.1 About Lotus QuickPlace

The best way to get acquainted with QuickPlace is to try it out. You can create your own test QuickPlace at the following Web address:

```
http://www.quickplace.com
```

Some of the things QuickPlace provides out of the box are:

Instant Creation—A secure Web workspace can instantly be created for your team—the startup time is literally 60 seconds.

Shared Content—Your team's discussions and documents are all maintained in a single location. Rich content can be created within or imported from Microsoft Office and other applications. Multiple revisions are easily tracked.

Shared Sense of Time—Gantt charts and calendars reflecting assigned tasks and meetings are available to help move the team toward its ultimate goals. Newsletters are e-mailed daily, providing an active pulse for the team. Real-time chat facilities are provided for instant contact.

Shared Process and Identity—As your project develops, the team develops a structure and process to meet its goals. The team's QuickPlace workspace easily adapts—"inner rooms" are available for sub-team security; browser-designed custom forms with approval cycles can be easily added; and the cosmetics of the QuickPlace can be quickly changed to reflect the team's emerging identity.

Shared Knowledge—When the project or initiative is over, a new one is just around the corner. Only QuickPlace allows you to capture all the previously applied structure and knowledge gained into a “PlaceType” solution module. PlaceTypes can be used as the basis for the next project/initiative by the same team or by other teams within your organization.

Deep Customization—QuickPlace is designed to be easy to use and productive immediately out of the box, but for the corporate developer, Lotus Business Partner or Application Service Provider, QuickPlace provides a wealth of customization opportunities, including browser customizations, HTML design, Java agents (PlaceBots) and server extensions. The gradient of customization extends QuickPlace from an “instant collaboration application” to an “instant collaboration platform.”

Directory Integration—Domino Name and Address, LDAP directory (MS Exchange, NT Domain) and automatic IIS detection and install option.

Microsoft Office 2000 Integration—Ability to author content from any Office application; import templates as customized forms.

Task Management—Assign action items, track status and monitor progress.

On-line Awareness and Chat—Brainstorm new ideas or review content with on-line team members.

Themes and PlaceType Solution Modules—Customize a QuickPlace workspace to mirror an existing intranet or corporate Web site; create specific solution modules to address critical business issues that require immediate response and execution.

Although this is only a brief overview of what Lotus QuickPlace is, it should be enough to get you started with the rest of this redbook. If you are interested in a more elaborate discussion of what QuickPlace is, refer to Appendix K, “Zen and the art of QuickPlace customization” on page 403.

1.1.2 Terminology used in this redbook

In this redbook you may encounter several new terms. Refer back to this list during your reading.

QuickPlace server	A Web-based groupware development platform to support team communications, unstructured work practices, content management, process management, and collaboration

Place	A virtual collaborative space, created on a QuickPlace server, instantiated as a group of Domino databases with “awareness of each other.” Sometimes we also refer to a Place as a QuickPlace, where it cannot be confused with the full QuickPlace server.
PlaceType	A design specification that gives an instantiated Place all of its default and configurable properties, including its appearance, structure, business logic, content, and programmatic behaviors; a QuickPlace server can be configured to allow the creation of a PlaceType from every Place, in order to reuse form, content, and processes.
PlaceBot	A programatic enhancement to a QuickPlace form, based on LotusScript or Java, that extends or alters the QuickPlace’s form handling behavior. PlaceBots are in essence similar to Domino agents, with the difference that Domino Designer is required to write an agent, while PlaceBots can be written in a simple text file and uploaded to the QuickPlace server from a Place.
Workflow	Applications or features that coordinate the work of groups, when appropriate, making sure that the decisions of one person are acted upon before being passed on to the next person; often embodying features such as privacy control, sequencing, notification, and routing.
Theme	A collection of elements that use the QuickPlace Layout Architecture to control the look and layout of a QuickPlace, often associated with the identity, purpose, and norms of the group. It is sometimes referred to as Skin.
Layout file	An HTML-formatted element of the QuickPlace Layout Architecture that defines the structure and layout of a Place’s design elements. It is sometimes also referred to as a Skin file.
Turnkey Server	A ready-to-use installation of a QuickPlace server, bundled with PlaceTypes customized to support targeted communities, teams, events, and processes; the Turnkey Server may also include additional application layers to enhance the enterprise integration and administration of QuickPlace, such as knowledge management or Web portal integration

1.2 Summary

In this chapter we have given a brief overview of What QuickPlace is, who this redbook is written for, and the terminology used in the book.

Chapter 2. Why QuickPlace customization

In this chapter we discuss why people, from users to programmers, want to customize QuickPlace, and we give an overview of the different ways to customize QuickPlace. In the rest of the book, we go into more detail about the customizations you can perform as a developer. Before leaving this chapter, we introduce a scenario that we will use in our examples throughout the rest of this book.

2.1 The motivation for customization

You can categorize the main drivers for customizing QuickPlace as belonging to one of the following three groups:

- **Extend** QuickPlace's already robust collaboration, document management, workflow, and administration capabilities
- **Integrate** QuickPlace with other applications and platforms
- **Reuse** components developed in or for QuickPlace and even reuse Places themselves with PlaceTypes

Throughout this redbook we give many examples of customization in each of these groups. But before we dive into the code, we first give an overview of the different ways to customize QuickPlace.

Note: For a more in-depth discussion of what motives there are for customization and examples of the kind of customizations performed by users, Place managers, Server administrators and developers, see Appendix K, "Zen and the art of QuickPlace customization" on page 403.

2.2 The different ways to customize QuickPlace

QuickPlace is by far the most customizable instant collaboration platform available for organizations engaging in collaborative projects or ad hoc initiatives. While the "out-of-the-box" capabilities of QuickPlace greatly enhance a team's performance, it is the customization capabilities of QuickPlace that allow advanced collaboration systems to evolve along with a team's unique mission.

2.2.1 Multiple audiences

QuickPlace allows you to customize for multiple audiences. Table 1 lists four different ways you can customize QuickPlace, and shows which QuickPlace elements you work with in each area.

Table 1. the QuickPlace Application Model

Structure Rooms Folders Forms	**Reuse** PlaceTypes Custom Install
Look and Feel Decorations Themes	**Logic** PlaceBots Workflow

From the table, you can see that QuickPlace provides a gradient of customization capabilities:

- **Browser-based customization**. No programming or special design tools are involved—you can customize the appearance, organization and functionality of your QuickPlace workspace simply by using your browser.
- **HTML-based customization**. Using any HTML authoring tools, you can develop special forms for QuickPlace. In addition, the entire user interface of QuickPlace is easily modified with new HTML “Themes” (skin) technology.
- **Java-based customization**. For programmers, QuickPlace offers robust programmable access to every aspect of the QuickPlace system, as well as integration with external databases and file systems.
- **DLL extensions**. For ultimate control, QuickPlace allows C programmers to “hook” key QuickPlace events, allowing DLL extensions to be added at the operating system level of the QuickPlace server.
- **Turnkey servers**. IT professionals and business partners may use the above capabilities in concert with the QuickPlace turnkey systems to develop Turnkey Servers for internal use or resale.

QuickPlace Release 2.0 is more than a product, it is a platform for the development and deployment of instant collaborative applications.

2.2.2 Customization—browser users

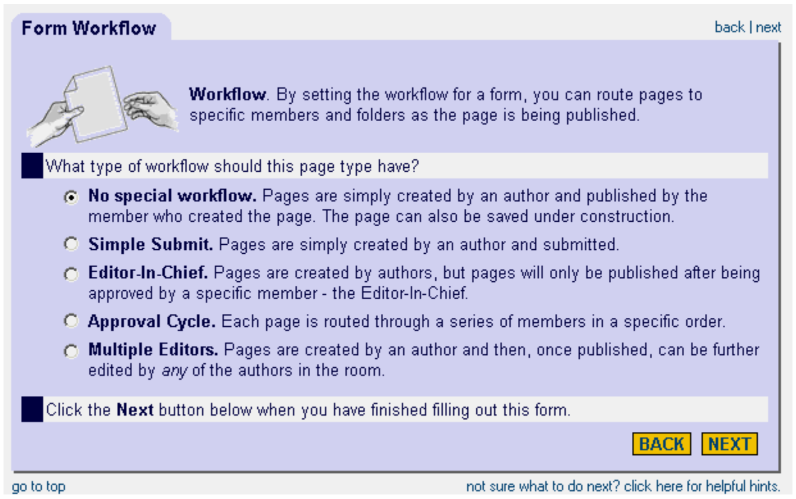
QuickPlace Release 2.0 supports a tremendous amount of customization that you can apply simply by using your browser.

2.2.2.1 Build custom forms

As mentioned earlier, you may build custom forms in a QuickPlace with your browser. The custom form is automatically laid out as you pick the type of fields you would like to include on the form.

2.2.2.2 Add simple Workflow to your forms

In the process of building a form, you may choose a workflow through which pages created with the form must travel, as shown in Figure 1.



Workflow. By setting the workflow for a form, you can route pages to specific members and folders as the page is being published.

What type of workflow should this page type have?

- [✓ **No special workflow.**] Pages are simply created by an author and published by the member who created the page. The page can also be saved under construction.
- **Simple Submit.** Pages are simply created by an author and submitted.
- **Editor-In-Chief.** Pages are created by authors, but pages will only be published after being approved by a specific member - the Editor-In-Chief.
- **Approval Cycle.** Each page is routed through a series of members in a specific order.
- **Multiple Editors.** Pages are created by an author and then, once published, can be further edited by *any* of the authors in the room.

Click the **Next** button below when you have finished filling out this form.

Figure 1. Specifying workflow for a type of pages

2.2.2.3 Build forms based on Microsoft Office files

You may also build forms based on any Microsoft Office document or file that you would like to use as a template. You can, for example, have a “boilerplate” Word file that you create for all customer proposals. Simply create an Office form that includes the boilerplate. New pages created with the form will automatically inherit the boilerplate file.

2.2.2.4 Customize the appearance of your QuickPlace workspace

You can dramatically modify the appearance of your QuickPlace workspace simply by filling out the Theme and Tweak forms offered in the QuickPlace customization area. You can choose Themes to make large sweeping changes to the overall look of your QuickPlace workspace.

Once a theme is selected, you can tweak the appearance of many aspects of your QuickPlace workspace by adding a logo, changing the background colors and font styles, etc. QuickPlace even has a special facility (that we call the Graphics server) that generates animated and specialty font graphics based on the text you provide.

None of the above customizations requires special software. They are achieved with your browser. These capabilities allow you and your team members to rapidly adapt your QuickPlace workspaces independently.

2.2.3 Customization—HTML designers

QuickPlace Release 2.0 supports a more advanced degree of customization for those who are comfortable authoring HTML with any HTML editor such as Macromedia Dreamweaver, Allaire HomeSite and others.

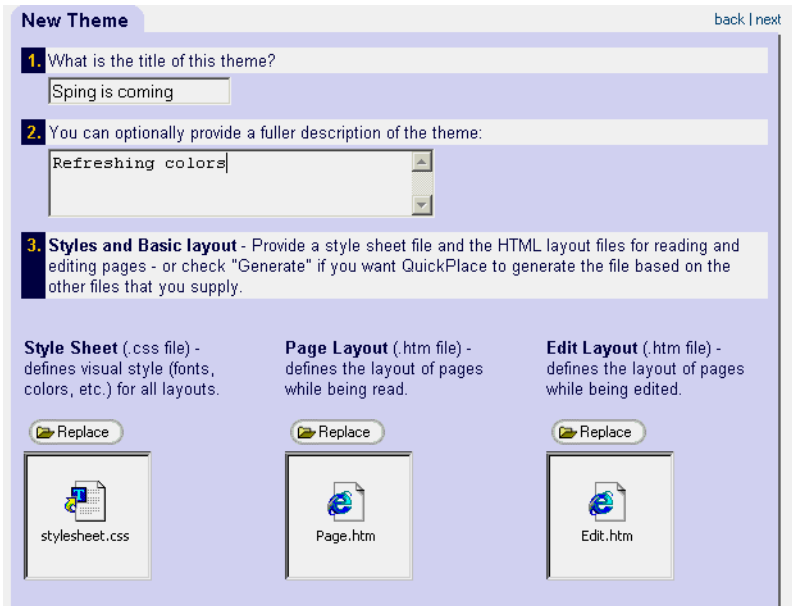
2.2.3.1 Import highly customized HTML forms into QuickPlace

You can import any HTML form into QuickPlace. The form can utilize all of the standard input fields, JavaScript, images, Flash files and more. The HTML form is imported through a custom import form in QuickPlace. QuickPlace automatically imports all associated resources (e.g., images), and even allows you to associate a workflow with the form. Once imported, QuickPlace will include your new form in the list of forms which team members can select to create new QuickPlace documents.

2.2.3.2 Change the complete layout with custom HTML Themes

You can completely change the appearance of your QuickPlace workspace by authoring your own custom HTML Theme (skin). A custom Theme is an HTML (or DHTML) file that references these QuickPlace UI elements: Search bar, Page title, Action buttons, and others. The UI elements are referenced through a special QuickPlace <insert Skin Component> tag.

You then upload the HTML Theme into your QuickPlace workspace as shown in Figure 2 on page 11.


New Theme

back | next

1. What is the title of this theme?

Sping is coming

2. You can optionally provide a fuller description of the theme:

Refreshing colors

3. **Styles and Basic layout** - Provide a style sheet file and the HTML layout files for reading and editing pages - or check "Generate" if you want QuickPlace to generate the file based on the other files that you supply.

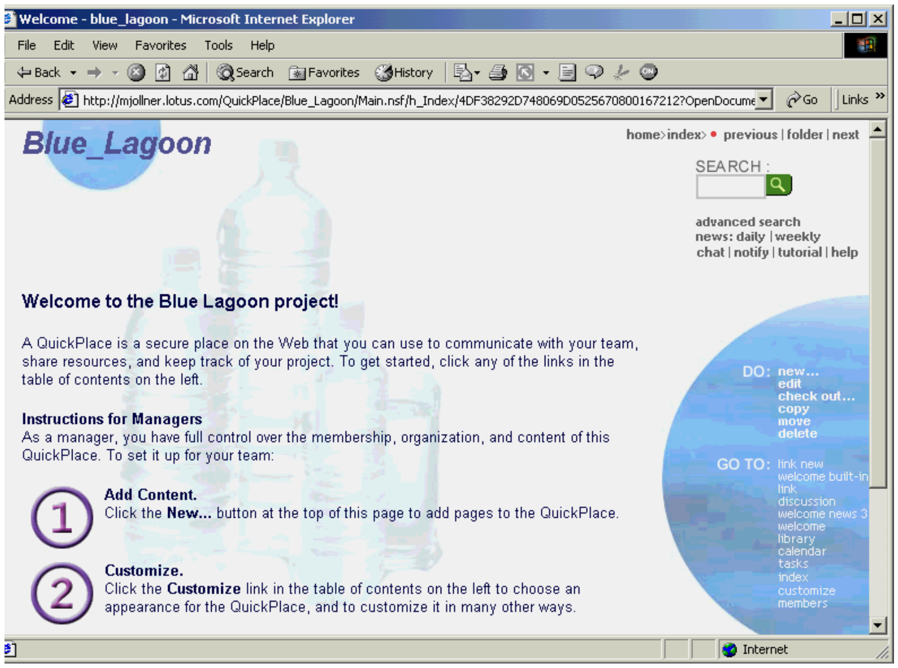
Style Sheet (.css file) - defines visual style (fonts, colors, etc.) for all layouts.

Page Layout (.htm file) - defines the layout of pages while being read.

Edit Layout (.htm file) - defines the layout of pages while being edited.

Figure 2. Uploading a Theme for QuickPlace

Once uploaded, QuickPlace adjusts every element within your QuickPlace workspace to reflect the new design, as we see in Figure 3 on page 12.


Welcome - blue_lagoon - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://mjollner.lotus.com/QuickPlace/Blue_Lagoon/Main.nsf/h_Index/4DF38292D748069D0525670800167212?OpenDocume
Go Links
Internet

Blue_Lagoon

home>index> • previous | folder | next

SEARCH :

advanced search
news: daily | weekly
chat | notify | tutorial | help

Welcome to the Blue Lagoon project!

A QuickPlace is a secure place on the Web that you can use to communicate with your team, share resources, and keep track of your project. To get started, click any of the links in the table of contents on the left.

Instructions for Managers
As a manager, you have full control over the membership, organization, and content of this QuickPlace. To set it up for your team:

1 **Add Content.**
Click the **New...** button at the top of this page to add pages to the QuickPlace.

2 **Customize.**
Click the **Customize** link in the table of contents on the left to choose an appearance for the QuickPlace, and to customize it in many other ways.

DO: new... edit check out... copy move delete

GO TO: link new welcome built-in link discussion welcome news 3 welcome library calendar tasks index customize members

Figure 3. QuickPlace Welcome page with new Theme applied

2.2.3.3 Use Themes for functional changes

You can use Themes to make broad functional changes in your QuickPlace workspace:

- Embed static legal/copyright information on every page to protect intellectual property.
- Embed critical links (URLs) to support staff or instructional materials that support the QuickPlace workspace.
- Embed links to custom sites that are affiliated with the QuickPlace workspace.
- Mirror the look of your QuickPlace workspace to a preexisting intranet portal or corporate Web site.
- Write a special Theme that takes unique advantage of IE5’s XML/XSL capabilities.

Themes truly allow you to exert brand and functional control over your QuickPlace workspace so that QuickPlace becomes part of a larger design, as opposed to a “spot solution.”

2.2.4 Customization—programmers

QuickPlace Release 2.0 supports unbounded customization for those of you who have Java, LotusScript or C programming skills.

2.2.4.1 Java PlaceBots

QuickPlace Release 2.0 provides a new PlaceBot (agent) capability that enables you to programmatically access every aspect of a QuickPlace workspace design and function.

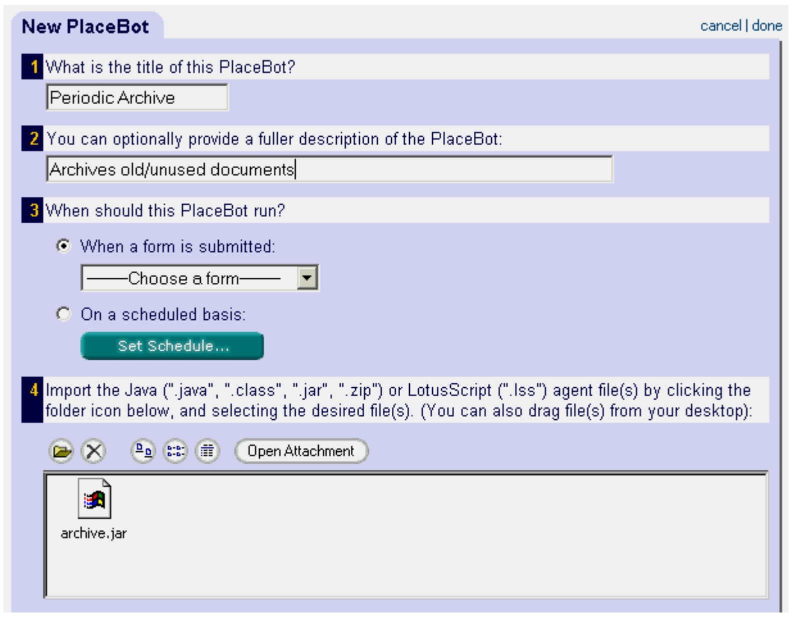
A PlaceBot is a procedure that is written in Java or LotusScript which manipulates QuickPlace data on a scheduled basis, or when a form is submitted within a QuickPlace. QuickPlace stores all its data in an object database. The QuickPlace object hierarchy that you may reference via Java or LotusScript methods is as follows:

- QuickPlace
- Room
- Folder
- Page
- Field-Item

The range of tasks that a PlaceBot can accomplish is truly unbounded. For example: A scheduled PlaceBot searches for documents where the Archive field is TRUE and moves those documents to a relational database off-site for safekeeping. A scheduled PlaceBot monitors disk utilization on a QuickPlace server and sends an e-mail notification to the server administrator when disk space is low.

- A form-submission PlaceBot tests the value of a *Customer Urgency* field within a form and, upon form submission, conditionally moves a new document into an *Urgent Requests* folder within the QuickPlace workspace.

The Java classes available to PlaceBot programmers extend far beyond the boundaries of the QuickPlace system, enabling integration with relational databases, Lotus Domino databases and file-based systems. PlaceBots may be written with any text editor and uploaded into a QuickPlace workspace via a form as shown in Figure 4 on page 14.


New PlaceBot

cancel | done

1 What is the title of this PlaceBot?

Periodic Archive

2 You can optionally provide a fuller description of the PlaceBot:

Archives old/unused documents

3 When should this PlaceBot run?

[✓ When a form is submitted:] ——Choose a form——

On a scheduled basis: Set Schedule...

4 Import the Java (".java", ".class", ".jar", ".zip") or LotusScript (".lss") agent file(s) by clicking the folder icon below, and selecting the desired file(s). (You can also drag file(s) from your desktop):

Figure 4. QuickPlace page used to define a PlaceBot

If you are experienced in developing Lotus Domino agents, a PlaceBot is best understood as an agent with an extended set of objects (e.g., rooms, members, etc.) that are meaningful in the context of QuickPlace Release 2.0.

2.2.4.2 DLL extensions for C programmers

C programmers may trap critical QuickPlace events and execute a server-side DLL prior to the event's execution within QuickPlace. Events include:

- Place Creation/Deletion
- Room Creation/Deletion
- Member Addition/Removal
- Page Creation/Deletion

Examples

You can use the Add Member DLL hook to integrate an external charge-back billing system that charges departments per QuickPlace member.

You can use the Create Place and Create Room DLL hooks to copy specialized forms from a centrally maintained “forms bin” into each new QuickPlace or QuickPlace room.

In summary, the range of customizations available to corporate developers and business partners is limited only by the imagination. It is this capacity that extends QuickPlace from an “instant collaboration application” to an “instant collaboration platform.”

2.2.4.3 Customization—Turnkey servers

QuickPlace Release 2.0 allows corporate developers or business partners to encapsulate customized QuickPlaces as Turnkey servers that can be widely distributed.

2.2.4.4 PlaceType Solution Modules

Once you have customized a QuickPlace workspace, you can simply declare it a PlaceType that is capable of redistribution. All the customizations (simple and advanced) that have been added to the QuickPlace workspace are encapsulated within a PlaceType.

Customizations include:

- Content
- Simple forms
- Folders/rooms
- Custom HTML forms
- Custom Themes
- PlaceBots
- DLL enhancements
- PlaceType profile

A PlaceType Solution Module, therefore, serves as a template for defining new QuickPlace workspaces that are best tuned to the needs of various projects, including the following:

- Product Development
- Emergency Response System
- Product Launch
- Consultancy Engagement
- Customer Service
- Task Force

2.2.4.5 Using a PlaceType solution module

PlaceType solution modules are maintained in a special directory on each QuickPlace server. When a user connects to a QuickPlace server to create a new QuickPlace workspace, that user is presented with a gallery consisting of all the PlaceTypes that are available on that server.

Once a PlaceType is selected from the gallery, a new QuickPlace workspace is created, inheriting all of the properties and content of the original PlaceType.

Active update. As time passes, QuickPlace workspace works that are based on a PlaceType may optionally be updated with enhancements that have been applied to the master QuickPlace workspace.

2.2.4.6 PlaceType solution module profile

With PlaceTypes, you can completely control how customizable a QuickPlace workspace that is created from a PlaceType will be by checking the features that you want to be available to the new QuickPlace. The PlaceType solution module profile also controls how the PlaceType is presented in the PlaceType gallery.

2.2.4.7 Creating a custom Turnkey server

As a corporate developer or business partner, you can remove any of the default PlaceTypes that Lotus provides and replace them with one or more PlaceTypes that are dedicated to your particular business application. A consulting firm, for example, may construct a server containing a single customized consultancy PlaceType that best serves the collaborative processes associated with consultants who manage multiple clients. The consultancy PlaceType might include the following customizations:

- A custom Theme that matches the brand of the consultancy's firm
- Links in the custom Theme that refer back to the consultancy's static Web site, as well as to special instructional material that is presented via RealVideo on a central server
- Standard documents and forms that apply to all consultancy situations
- A room structure that supports preparing documents privately prior to revealing them to the clients

A consultant who connects to this QuickPlace server will perceive that he or she has connected to a *Consultancy Collaboration Server*. Note that you may completely replace the home page of the QuickPlace server with an HTML page containing special URLs that, when clicked, will create a QuickPlace of *a specific PlaceType.*

2.2.4.8 Reselling PlaceTypes as a Turnkey Server or service

Finally, as a QuickPlace Business Partner, you may resell QuickPlace Turnkey Servers into vertical markets. You may modify the images that appear during server installation so that your customers install your complete solution.

If you are an Application Service Provider (ASP), you may also host QuickPlace Turnkey servers to provide highly tuned instant collaboration services to vertical markets.

2.3 Our redbook scenario - Millennia Consulting

As a vehicle to understanding customizing QuickPlace in the context of business operations, we will consider a hypothetical company that uses QuickPlace to deliver consulting services (and consulting services to deliver QuickPlace).

2.3.1 A solutions-based approach

Just as applications created to solve particular problems require customization, larger-scale deployments often need customization to fit into the bigger picture of enterprise applications. These applications are so tightly intertwined with the organization’s critical processes that they often require a solutions-based approach, in which the technological tool is embedded in process improvement, training, and management consulting. We will demonstrate a solutions-based approach in which customization will be an ongoing cycle of requirements analysis, design, rapid-prototyping, tweaking, and re-design of another reusable, and perhaps more specialized, prototype.

2.3.2 Scenario background

Millennia Merger & Acquisitions Services is a thriving practice at Millennia Consulting, one of the nation's leading professional services firms which provides assurance advisory, tax and management consulting services through more than 30,000 people in over 100 U.S. cities.

Millennia's deep resources give them (and their customers) a competitive advantage. Millennia's professional advisors are committed to providing:

- Responsiveness
- Creative ideas for adding value, thereby helping you increase your investment returns
- Effective communications with customers and other advisors

- The ability to draw from industry and technical specialists in order to field just the right multifunctional team

2.3.3 Scenario

Millennia Mergers & Acquisitions services division has been retained by a large software company—TheRock, Inc.—to manage the acquisition of a 30-person Internet startup—CapMan Software. TheRock is a publicly traded U.S.-based company, while CapMan is a privately held company based in Sydney, Australia.

The desire to close this deal is mutual and so far dealings have been amicable. A letter of intent to purchase has been sent to CapMan's president, Livingstone Campbell, by TheRock legal team after initial in-person meetings. Millennia Consulting's M&A team is in place and ready to facilitate each phase of the acquisition process:

- *Creative Tax and Accounting Structuring*
- *Due Diligence Assistance*
- *Assistance with Acquisition and Financing Documents*
- *Cross-Border Interests*
- *Closing Assistance*
- *Post-Transaction Assistance*

2.3.4 Millennia needs a virtual place to bring them together

Fortunately, Millennia has just deployed a family of QuickPlace-based Solutions to manage projects and improve business communication. It's believed these new solutions will help Millennia exceed client expectations and provide more effective and efficient communications between Millennia Consulting Practitioners and their clients. The high-level requirements of these solutions are:

- An extranet solution, to include team members from TheRock and CapMan
- Convey a sense of team and collaboration—community
- Easy-to-manage content
- Re-usable tools
- Integration with the Document Management system
- High security

More specific requirements will be mentioned with the examples where they are relevant.

2.3.4.1 Examples mapped to the scenario

To give you an idea of the kind of examples in the book, we will mention some of them here. The examples are covered in the different chapters in the book, but we will refer to these examples again when we discuss PlaceTypes and Turnkey servers in 11.5, “Adding structure to a new Place - a scenario” on page 331. Note that there are many other examples in the redbook as well.

QuickPlace Creation Management

This example will show how to create a simple front-end for the QuickPlace server that will provide a vehicle for automating the managed creation of new QuickPlaces by presenting a link to submit a request for a new QuickPlace. A simple workflow process will be incorporated, along with discussion about how you might handle more sophisticated processes using Domino Workflow.

QuickPlace Server Administrative Reporting (TheXRay)

In this example we will show how to create reporting tools for administrators to monitor the QuickPlace server. The statistical data will be collected and compiled into graphic representations ,making it easy for the administrator to “take the temperature” of the QuickPlace server.

QuickPlace content life cycle management

Here we will show the benefit of integrating QuickPlace with Domino.Doc by leveraging the strengths of each product. Content creation in QuickPlace is a more “organic” creation model suitable for teaming. Domino.Doc has a more structured storage model that is suitable for life cycle management. For the example, we show you how to move selected content that was created in QuickPlace at determined intervals to a Domino.Doc library in order to benefit from the document management provisions in Domno.Doc.

Automatic site map creation

This example will show you how to create self-maintaining site (QuickPlace) mapping for the users of a QuickPlace. This method provides easy and fast access to content within a QuickPlace. We will use PlaceBots to perform the automated creation of the mapping.

QuickPlace mail room attendant

Here we will show you how to create a PlaceBot that will use special rooms to manage incoming mail to the QuickPlace. This method has proven useful for handling news feeds that have been set up to automatically send content to the QuickPlace. The Mail Room Attendant can be trained to distribute incoming mail to specified rooms within the QuickPlace to help organize this type of content automatically.

Note: We have used the Millennia Consulting scenario as a red thread for the examples we have developed throughout this book. We do not claim in any way to have a full functional solution for a company like Millennia Consulting, but we illustrate the *techniques* necessary for developing such total solutions.

2.4 Summary

In this chapter we have described what drives the need for QuickPlace customization. We also briefly examined the different areas where you can customize QuickPlace, and described how to do it. In the last part of the chapter we introduced the scenario we will bind most of our examples to, and we listed some of the examples you will find in this redbook.

Chapter 3. Installing the QuickPlace server

The QuickPlace server can be installed as a standalone server, or it can be installed on top of an existing Domino server. In standalone mode, almost all the configuration a QuickPlace server requires can be accomplished via a browser. When running it on top of a Domino server, depending upon your QuickPlace and Domino configurations, some changes may also be required in the Domino server document.

This chapter supplements the information in installation guide and user's guide for the QuickPlace server. For the latest release notes visit QuickPlace DevZone at:

`http://www.quickplace.com/devzone`

3.1 Installation

This section describes how to install a QuickPlace server. Installation varies depending upon the host operating system. We describe two ways to install QuickPlace:

- standalone installation
- Overlay installation on an existing Lotus Domino server

Here, our focus is on Windows NT. For the other platforms refer to their respective installation guides. Some AS/400-related information is in Appendix B, "QuickPlace for AS/400 considerations" on page 359.

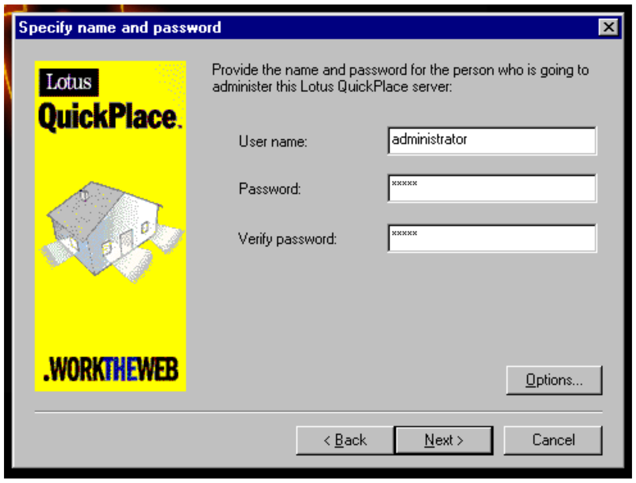
3.1.1 Standalone QuickPlace server

In standalone mode, the QuickPlace server does not need a pre-installed Domino server. Although it still uses a Domino engine for the services, the minimal Domino code is part of the standalone installation.

Run the setup.exe that is part of the QuickPlace installation package. Select the appropriate drive and directory (the default is c:\lotus\quickplace) where you want to install the software.

You are then prompted for the administrator's user ID and password (see Figure 5 on page 22). This is also the place where you can optionally supply the certifier information.

Note: This is important if your organization has an existing Domino environment and you want to customize the QuickPlace environment. Clicking the **Options** button lets you provide the same certifier information for QuickPlace and Domino (Figure 6). Read more about this in 3.1.3, “Certifier ID considerations and certifier hierarchy” on page 25.



Specify name and password

Provide the name and password for the person who is going to administer this Lotus QuickPlace server:

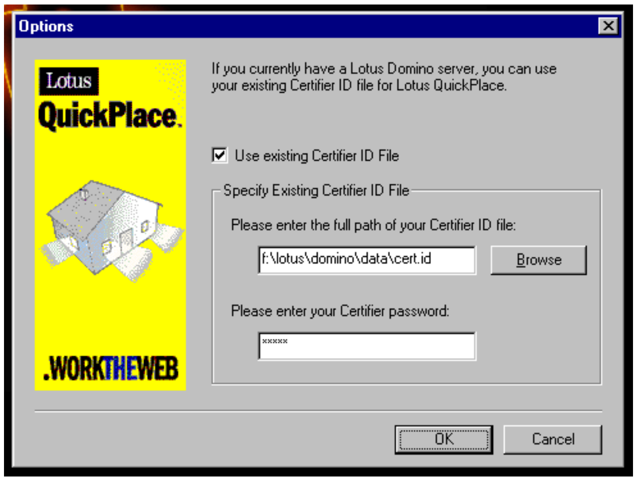
User name: administrator

Password: *****

Verify password: *****

Figure 5. Specify name and password

You must write the user ID and password down and keep it in a safe place in case you forget.



Options

If you currently have a Lotus Domino server, you can use your existing Certifier ID file for Lotus QuickPlace.

☑ Use existing Certifier ID File

Specify Existing Certifier ID File

Please enter the full path of your Certifier ID file:

f:\lotus\domino\data\cert.id

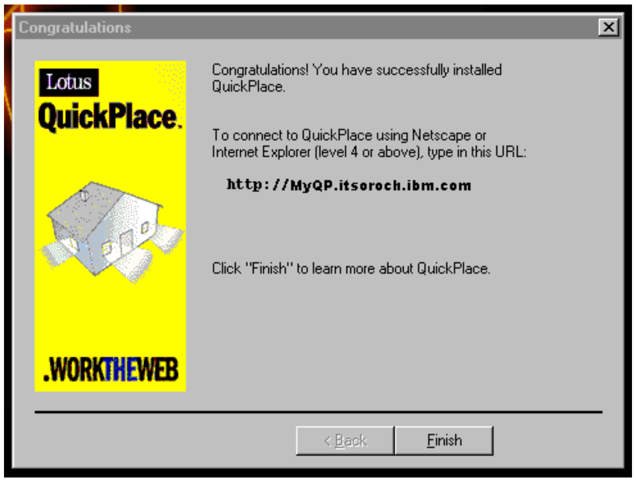
Please enter your Certifier password:

*****

Figure 6. Certifier information

The installation process picks up all the network information (like the host name, domain, TCP/IP address, etc.) from the network configuration of the Windows NT server you are installing from.

For us, the host name was *MyQP* (which is the same as our machine name) and the domain was *itsoroch.ibm.com*. Once the installation was successful (see Figure 7), the preferred browser was automatically launched with the readme.htm file that contained a link to our newly installed QuickPlace site. It also provided information on how to start and stop the QuickPlace server via the Services dialog under the Control Panel in Windows NT.



Congratulations! You have successfully installed QuickPlace.

To connect to QuickPlace using Netscape or Internet Explorer (level 4 or above), type in this URL:

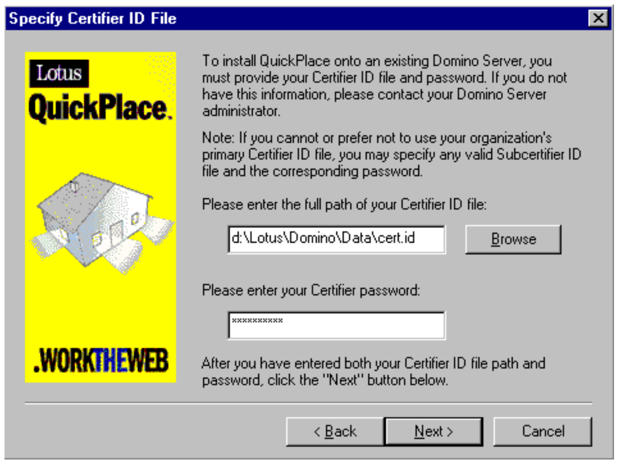
```
http://MyQP.itsoroch.ibm.com
```

Click "Finish" to learn more about QuickPlace.

Figure 7. Successful QuickPlace installation

3.1.2 QuickPlace server on Domino 5.x

Installing a QuickPlace server on top of an existing Domino server is similar to the standalone version, but there are a few differences. One difference is that the install process requires that you provide the certified ID file, along with the password, that your existing Domino environment is using (see Figure 8 on page 24).



To install QuickPlace onto an existing Domino Server, you must provide your Certifier ID file and password. If you do not have this information, please contact your Domino Server administrator.

Note: If you cannot or prefer not to use your organization's primary Certifier ID file, you may specify any valid Subcertifier ID file and the corresponding password.

Please enter the full path of your Certifier ID file:

d:\Lotus\Domino\Data\cert.id

Please enter your Certifier password:

****

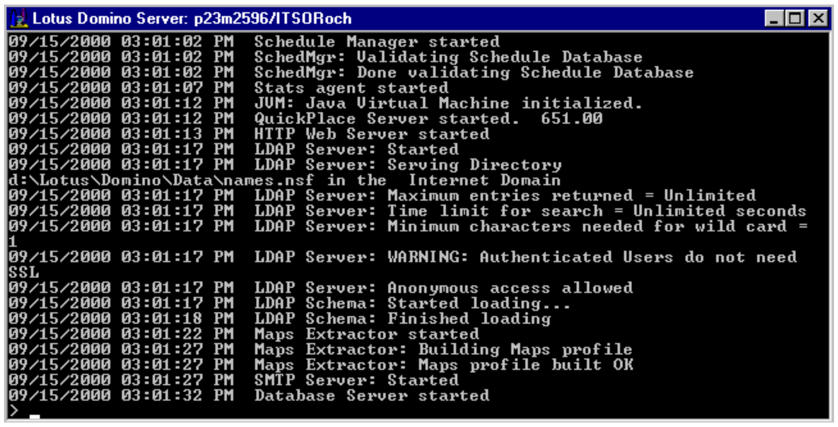
After you have entered both your Certifier ID file path and password, click the "Next" button below.

Figure 8. Request to supply certifier ID information

Once the installation is complete, the readme file is displayed in the browser. As in the standalone mode, QuickPlace server is automatically launched - in this case the Domino server is started. Figure 9 on page 25 shows the start of the QuickPlace process being logged in the Domino console.

The Domino server should not be running when installing QuickPlace, otherwise an error is raised and installation is terminated. This mode of QuickPlace installation is also referred to as an *overlay*. In this mode, the installation process modifies only two Domino files: Notes.ini and Domino Directory (names.nsf). All the other files, which are modified or copied, are QuickPlace server-specific.

For QuickPlace 2.0 version, Domino 5.0.4 release is required. For QuickPlace 2.05 version, Domino 5.05 release is required.



```
09/15/2000 03:01:02 PM  Schedule Manager started
09/15/2000 03:01:02 PM  SchedMgr: Validating Schedule Database
09/15/2000 03:01:02 PM  SchedMgr: Done validating Schedule Database
09/15/2000 03:01:07 PM  Stats agent started
09/15/2000 03:01:12 PM  JVM: Java Virtual Machine initialized.
09/15/2000 03:01:12 PM  QuickPlace Server started.  651.00
09/15/2000 03:01:13 PM  HTTP Web Server started
09/15/2000 03:01:17 PM  LDAP Server: Started
09/15/2000 03:01:17 PM  LDAP Server: Serving Directory
d:\Lotus\Domino\Data\names.nsf in the  Internet Domain
09/15/2000 03:01:17 PM  LDAP Server: Maximum entries returned = Unlimited
09/15/2000 03:01:17 PM  LDAP Server: Time limit for search = Unlimited seconds
09/15/2000 03:01:17 PM  LDAP Server: Minimum characters needed for wild card =
1
09/15/2000 03:01:17 PM  LDAP Server: WARNING: Authenticated Users do not need
SSL
09/15/2000 03:01:17 PM  LDAP Server: Anonymous access allowed
09/15/2000 03:01:17 PM  LDAP Schema: Started loading...
09/15/2000 03:01:18 PM  LDAP Schema: Finished loading
09/15/2000 03:01:22 PM  Maps Extractor started
09/15/2000 03:01:27 PM  Maps Extractor: Building Maps profile
09/15/2000 03:01:27 PM  Maps Extractor: Maps profile built OK
09/15/2000 03:01:27 PM  SMTP Server: Started
09/15/2000 03:01:32 PM  Database Server started
>
```

Figure 9. QuickPlace process in Domino console

3.1.3 Certifier ID considerations and certifier hierarchy

If you plan to profoundly customize QuickPlace environment to enhance its capabilities, or to suit your specific requirements (like creating special PlaceBots called *agents* to be executed on certain events, or creating special forms or views that may require the use of Domino Designer), then the IDs used to sign the design elements need special consideration.

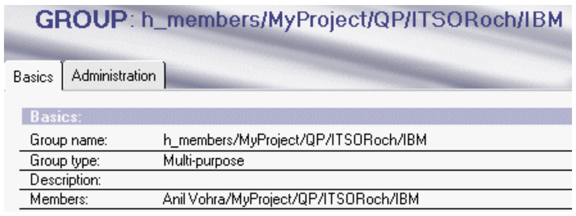
Domino is a *security-rich* environment. It also follows a hierarchical naming convention. As a Domino administrator you can control which users, group of users, organization units, and organizations can perform certain operations. The certifier ID file is what forms the base for the hierarchy. Your application designers can then use the appropriate hierarchy certificate (certifier) while building the design elements like forms, agents, etc. so as to avoid any security violations in the QuickPlace server.

When using standalone QuickPlace server, you can enter the certifier ID and password in the advanced section. Contact your Domino administrator to obtain the relevant certifier ID file and its password. If you need to customize the QuickPlace environment using the Domino Designer, then be sure to *specify the same certifier ID that is used by your development team*.

The Domino IDs that the designers use to create agents should have valid execution rights on the Domino server. You can specify this in the server document of the Domino Directory (names.nsf) under the *Agents Restrictions* section of the **Security** tab. Alternatively, you may sign the database (or elements) with the Domino server ID (Quickplace server ID in the standalone mode). During the QuickPlace server installation, make sure that the certifier file you provide is the one that is base for the IDs that you or your developers use to sign the design elements.

Refer to the Domino Administration and Domino Designer documentation for additional details about the certifier ID file.

Note: The certifier ID file may not contain more than two OUs. QuickPlace uses two additional OUs - one always named QP and another has the same name as that of the QuickPlace. For example, in our case the certifier ID had OU=ITSORoch/o=IBM as the hierarchy. So, the manager of our newly created QuickPlace (MyProject) had the canonical name, as shown in the Members field of Figure 10, within the QuickPlace environment.

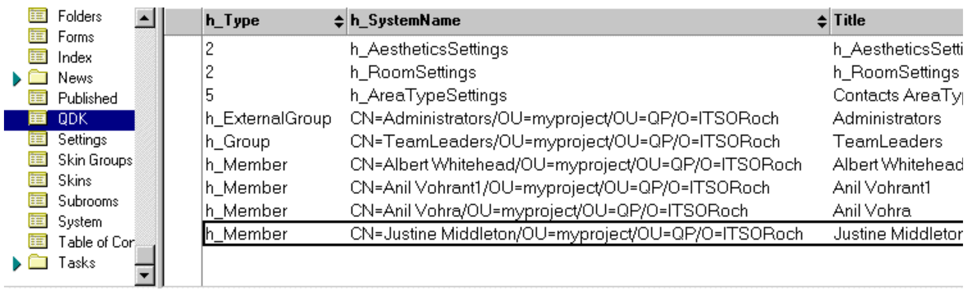


Group name:	h_members/MyProject/QP/ITSORoch/IBM
Group type:	Multi-purpose
Description:	
Members:	Anil Vohra/MyProject/QP/ITSORoch/IBM

Figure 10. Organization hierarchy in QuickPlace

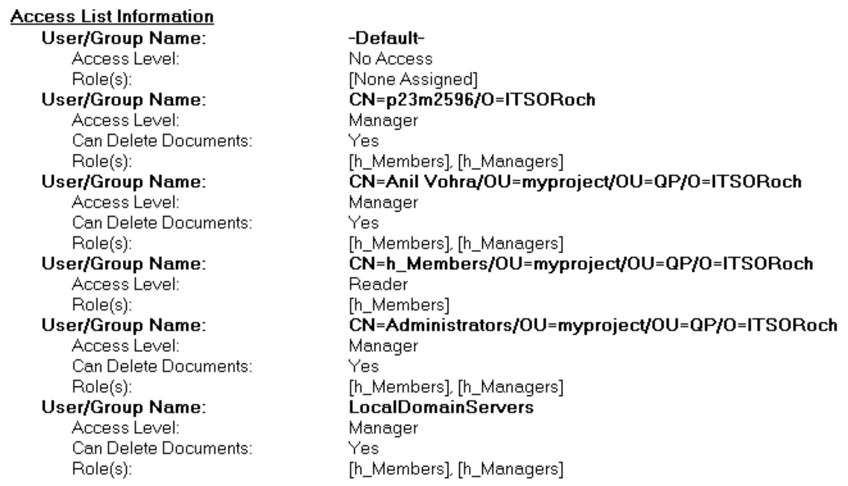
For every QuickPlace, h_members is the group (with the proper hierarchy) that is created in the Domino Directory (names.nsf). This is where all the members of a particular Place are listed. Contacts1.nsf (Member List) of every Place contains person records of all the members (see Figure 11).

The group, h_members, is used in the ACL of Contacts1.nsf to limit read access to the members only (see Figure 12 on page 27). The ACL of main.nsf controls the access to that Place. The managers for the Place are listed with Manager access in the ACL, and so are the authors and readers with corresponding ACL rights (see Figure 13 on page 27).



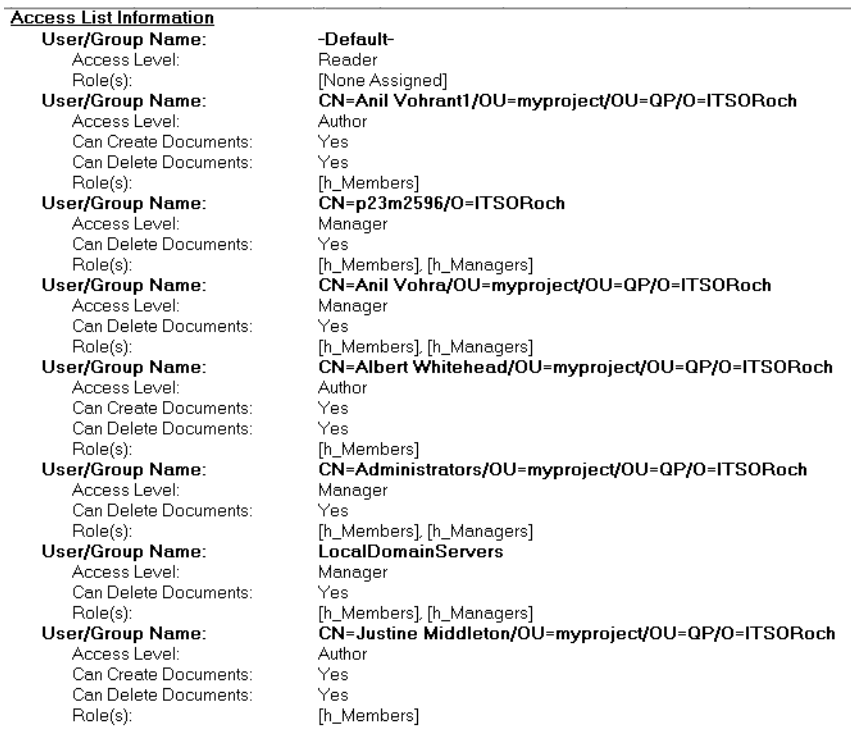
h_Type	h_SystemName	Title
2	h_AestheticsSettings	h_AestheticsSetti
2	h_RoomSettings	h_RoomSettings
5	h_AreaTypeSettings	Contacts AreaTy
h_ExternalGroup	CN=Administrators/OU=myproject/OU=QP/O=ITSORoch	Administrators
h_Group	CN=TeamLeaders/OU=myproject/OU=QP/O=ITSORoch	TeamLeaders
h_Member	CN=Albert Whitehead/OU=myproject/OU=QP/O=ITSORoch	Albert Whitehead
h_Member	CN=Anil Vohrant1/OU=myproject/OU=QP/O=ITSORoch	Anil Vohrant1
h_Member	CN=Anil Vohra/OU=myproject/OU=QP/O=ITSORoch	Anil Vohra
h_Member	CN=Justine Middleton/OU=myproject/OU=QP/O=ITSORoch	Justine Middletor

Figure 11. QDK view of Contacts1.nsf (MyProject) with all the members and local group



```
Access List Information
    User/Group Name:              -Default-
        Access Level:             No Access
        Role(s):                  [None Assigned]
    User/Group Name:              CN=p23m2596/O=ITSORoch
        Access Level:             Manager
        Can Delete Documents:     Yes
        Role(s):                  [h_Members], [h_Managers]
    User/Group Name:              CN=Anil Vohra/OU=myproject/OU=QP/O=ITSORoch
        Access Level:             Manager
        Can Delete Documents:     Yes
        Role(s):                  [h_Members], [h_Managers]
    User/Group Name:              CN=h_Members/OU=myproject/OU=QP/O=ITSORoch
        Access Level:             Reader
        Role(s):                  [h_Members]
    User/Group Name:              CN=Administrators/OU=myproject/OU=QP/O=ITSORoch
        Access Level:             Manager
        Can Delete Documents:     Yes
        Role(s):                  [h_Members], [h_Managers]
    User/Group Name:              LocalDomainServers
        Access Level:             Manager
        Can Delete Documents:     Yes
        Role(s):                  [h_Members], [h_Managers]
```

Figure 12. ACL details of Contacts1.nsf for the test (MyProject) Place



```
Access List Information
    User/Group Name:              -Default-
        Access Level:             Reader
        Role(s):                  [None Assigned]
    User/Group Name:              CN=Anil Vohrant1/OU=myproject/OU=QP/O=ITSORoch
        Access Level:             Author
        Can Create Documents:     Yes
        Can Delete Documents:     Yes
        Role(s):                  [h_Members]
    User/Group Name:              CN=p23m2596/O=ITSORoch
        Access Level:             Manager
        Can Delete Documents:     Yes
        Role(s):                  [h_Members], [h_Managers]
    User/Group Name:              CN=Anil Vohra/OU=myproject/OU=QP/O=ITSORoch
        Access Level:             Manager
        Can Delete Documents:     Yes
        Role(s):                  [h_Members], [h_Managers]
    User/Group Name:              CN=Albert Whitehead/OU=myproject/OU=QP/O=ITSORoch
        Access Level:             Author
        Can Create Documents:     Yes
        Can Delete Documents:     Yes
        Role(s):                  [h_Members]
    User/Group Name:              CN=Administrators/OU=myproject/OU=QP/O=ITSORoch
        Access Level:             Manager
        Can Delete Documents:     Yes
        Role(s):                  [h_Members], [h_Managers]
    User/Group Name:              LocalDomainServers
        Access Level:             Manager
        Can Delete Documents:     Yes
        Role(s):                  [h_Members], [h_Managers]
    User/Group Name:              CN=Justine Middleton/OU=myproject/OU=QP/O=ITSORoch
        Access Level:             Author
        Can Create Documents:     Yes
        Can Delete Documents:     Yes
        Role(s):                  [h_Members]
```

Figure 13. ACL details of Main.nsf for the test (MyProject) Place

You now have some idea of how the QuickPlace server uses the OUs and the ACLs, which should be helpful when manipulating the user names via an agent or program.

3.1.4 Standalone or overlay

When installing the QuickPlace Server, the question arises whether one should install the QuickPlace server in a standalone mode or as an overlay (on top of the Domino server). The simplistic answer is that if there is an existing Domino environment in the organization, then it is advantageous to install the QuickPlace server as an overlay. Of course, hardware and bandwidth capacity always need due consideration before reaching a conclusion.

The real benefit of installing the QuickPlace server on top of Domino (overlay) is in the area of management and logs. You can leverage all the existing Domino management tools, along with the administration and support processes; for example, you can continue the logging and analysis via domlog.nsf. If you have a data backup process in place for your Domino applications, the same process can automatically back up QuickPlaces. Also, in most cases, end-user support can piggy-back on the problem management model (help desk) for the existing applications.

A standalone out-of-box QuickPlace installation runs great and has no issues. But if you have an existing Lotus Domio environment and would like to customize or enhance the QuickPlace capabilities, then it is suggested that you install the QuickPlace server on top of a Domino server.

Disadvantages of running QuickPlace as an overlay may include:

- Additional load on the Domino server may impact the response times for existing Domino users.
- Any problem with the Domino server may also impact the QuickPlace users.

Keep in mind that, regardless of whether QuickPlace is installed in standalone mode or as an overlay, if Domino and QuickPlace are installed on the same piece of hardware, then any hardware trouble may impact both environments.

Notes

- QuickPlace is supported on a partitioned Domino server if you have an IP address for each of your Domino partitions. QuickPlace is not supported on a partitioned Domino server set up to use port mapping and to share a single IP address with other partitioned servers.
- QuickPlace server can be set up (standalone or overlay) to work with Microsoft's Internet Information Server (IIS) 4.0 and 5.0. For detailed instruction, visit the QuickPlace DevZone Web site at:

  `http://www.quickplace.com/devzone`

  Click **Release Notes**, and then **Installation**.

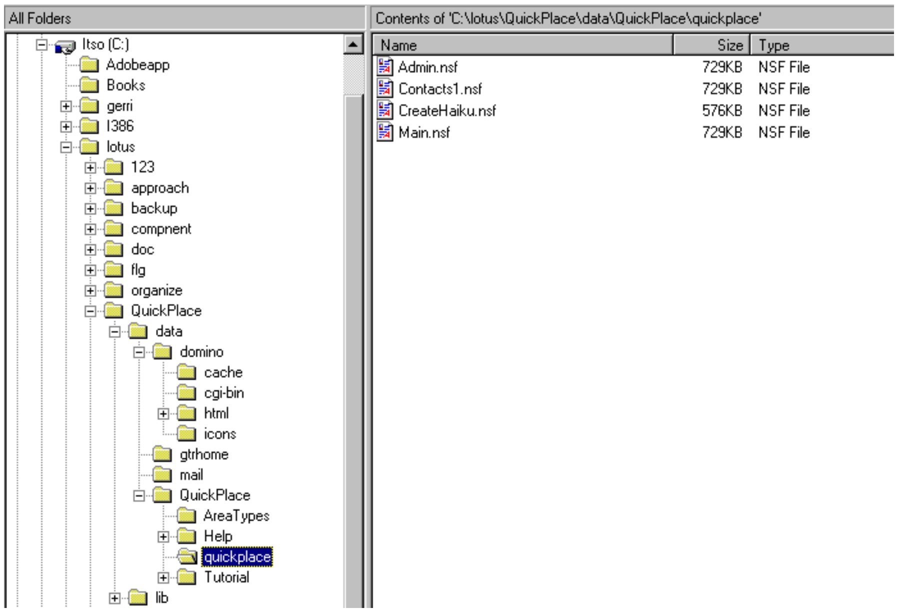
3.2 QuickPlace file system directory architecture

This section describes which directories, under the QuickPlace infrastructure, get created and what they contain.

3.2.1 QuickPlace file system directories

Let's first discuss the basics. Every new Place you create gets its own directory under the QuickPlace master directory. The name of that directory is same as the name of the Place. Each additional room in your Place is another file (.nsf) in your Place directory.

During QuickPlace server installation, the default QuickPlace (also called the "Welcome" QuickPlace) with the name of *QuickPlace* is automatically created. This is the entry point to the QuickPlace server, including server administration. Figure 14 on page 30 shows the directory structure, along with the files in "Welcome" QuickPlace, in a standalone QuickPlace server.



```
All Folders
Itso (C:)
    Adobeapp
    Books
    gerri
    I386
    lotus
        123
        approach
        backup
        compnent
        doc
        flg
        organize
        QuickPlace
            data
                domino
                    cache
                    cgi-bin
                    html
                    icons
                gtrhome
                mail
                QuickPlace
                    AreaTypes
                    Help
                    quickplace
                    Tutorial
            lib
```

Contents of 'C:\lotus\QuickPlace\data\QuickPlace\quickplace'

Name	Size	Type
Admin.nsf	729KB	NSF File
Contacts1.nsf	729KB	NSF File
CreateHaiku.nsf	576KB	NSF File
Main.nsf	729KB	NSF File

Figure 14. Directory structure of a standalone QuickPlace installation

For example if Millennia is our Place, then the basic infrastructure of the Millennia Place resides under \lotus\domino\data\quickplace\millennia (on the Domino server - given that \lotus\domino\data is the data directory) or \lotus\quickplace\data\quickplace\millennia (on standalone - given that \lotus\quickplace is the QuickPlace installation directory).

3.2.2 Administrator's Place

When QuickPlace server is installed, the "Welcome" region or the "Administrator's Place" is pre-configured to allow an entry point to the QuickPlace server. You as an administrator can then administer the newly installed QuickPlace server from this entry point.

This so-called administrative QuickPlace resides under the *QuickPlace* directory, which in turn is under the data directory. In our case, it was c:\lotus\domino\data\QuickPlace\QuickPlace when installed under Domino, and c:\lotus\QuickPlace\data\QuickPlace when in standalone mode. It contained the following files: Main.nsf; Contacts1.ns; CreateHaiku.nsf; Admin.nsf. The templates for these Domino databases reside in the directory named *AreaTypes*.

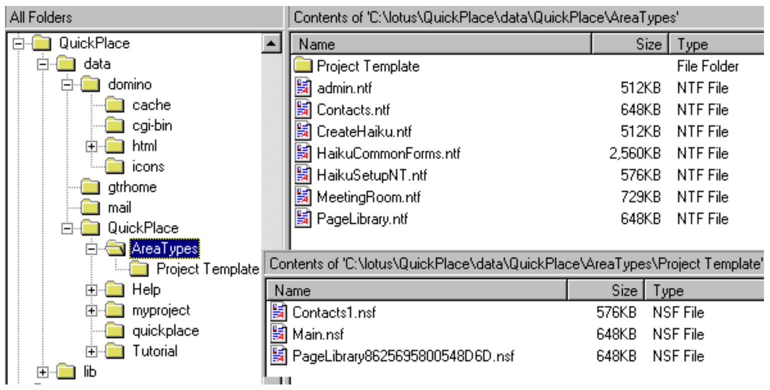
You can also tailor the "Welcome" page to suit your organization's needs.

3.2.3 Templates (AreaTypes or PlaceTypes)

After we created a few Places and used them for a while, we realized that some of the Places can be reused over and over again (for example, a Place to track a project). To avoid “reinventing the wheel” (that is, tailoring a Place from scratch), the popular Places can be saved as templates called PlaceTypes in the QuickPlace environment.This is discussed in more detail in 11.2, “Creating a PlaceType” on page 323.

We created a Place called MyProject. It had one room called “Status Meeting”. The room is what created the PageLibrary file. Each room resides in its own PageLibrary file.

As you can see in Figure 15, the newly created PlaceType gets its own directory under AreaTypes. Following is the directory structure:



```
All Folders
QuickPlace
  data
    domino
      cache
      cgi-bin
      html
      icons
    gtrhome
    mail
    QuickPlace
      AreaTypes
        Project Template
      Help
      myproject
      quickplace
      Tutorial
  lib
```

Contents of 'C:\lotus\QuickPlace\data\QuickPlace\AreaTypes'

Name	Size	Type
Project Template		File Folder
admin.ntf	512KB	NTF File
Contacts.ntf	648KB	NTF File
CreateHaiku.ntf	512KB	NTF File
HaikuCommonForms.ntf	2,560KB	NTF File
HaikuSetupNT.ntf	576KB	NTF File
MeetingRoom.ntf	729KB	NTF File
PageLibrary.ntf	648KB	NTF File

Contents of 'C:\lotus\QuickPlace\data\QuickPlace\AreaTypes\Project Template'

Name	Size	Type
Contacts1.nsf	576KB	NSF File
Main.nsf	648KB	NSF File
PageLibrary8625695800548D6D.nsf	648KB	NSF File

Figure 15. Directory structure example for “Project Template” PlaceType

Figure 15 also shows the default Domino template files (.ntf) that the QuickPlace server uses to create, when demanded by the end-users, all the subsequent Places. Help and Tutorial QuickPlaces which were created during the installation process also use some of the same templates.

3.2.4 Places

Places are the QuickPlaces that you or your users create on the server. Each QuickPlace gets a new directory under the <data>\QuickPlace directory. We used the default directory during a standalone QuickPlace server installation and that was c:\Lotus\QuickPlace\data. We created three Places: MyProject; NewProject; ShoreProject. Figure 16 on page 32 shows the directory hierarchy of the user-created Places.

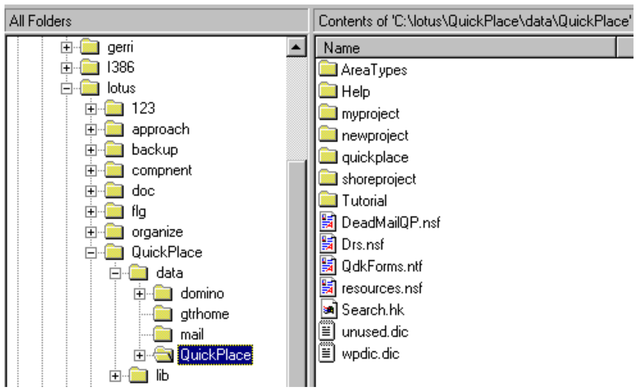


Figure 16. Where Places live

3.3 Security and access control options

In this section we provide a brief discussion of the security issue; for more detailed information, refer to *QuickPlace 2.0 User's Guide*, Chapter 5 "Administering a QuickPlace Server".

The security can be controlled at two levels:

- From the server's administration perspective - managing the QuickPlace server
- From each Place's perspective - managing a Place

As a server administrator, one can restrict who can create a new Place on the server and also who can administer the QuickPlace server. It is always better to have at least two people as administrators of any server. SSL encryption can also be controlled by the server administrator. SSL encryption is effective server-wide. There is no function to control it at a single Place level. All this can be accomplished via security screen.

Workaround to SSL-enable individual Places

If you have a Place where no new rooms are created, or you control the creation, there is a workaround that allows SSL enabling of individual Places. To enable SSL for a single Place:

1. Identify the Domino databases that make up the Place and open them in the Notes client. Select **File -> Database -> Properties.** The Properties box is shown.
2. On the first Tab (Database Basics), select the check box named **Web access: Require SSL connection**. Do this for *all databases in the Place*.

If a new room is added, the database created will not have this setting enabled by default, so you must make sure that you enable it manually before opening the room for business.

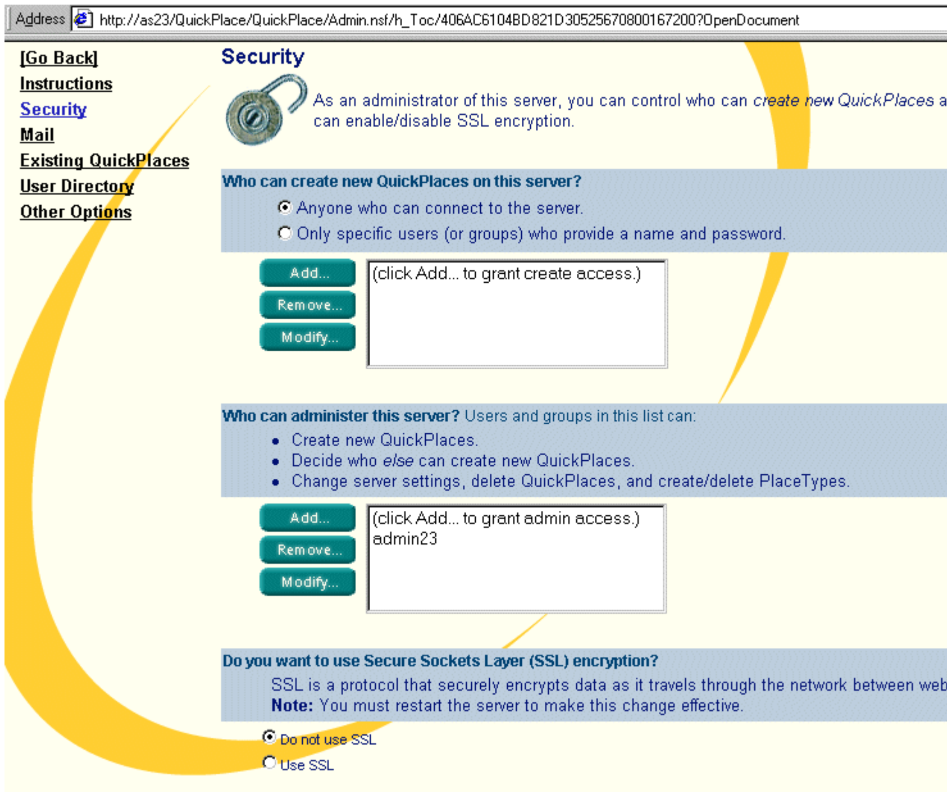
Now let’s go back to our discussion of the built-in QuickPlace security functions. At the Place level, you can control who can read the information, which users can create the information, and who can administer the particular Place. *Anonymous* allows everyone access without authentication.

If you already have a directory (Domino, LDAP or NT), you can link that to the QuickPlace server for user selection.

3.3.1 Managing the QuickPlace server

This is where you, as an administrator (and after signing in) manage the QuickPlace server. To get to the administration panel, go to the QuickPlace server’s main screen (also called the Welcome screen) and click **Sign In** in the top right-hand corner. Enter the user ID and the password of the administrator (typically the ID and password that was used during installation of the QuickPlace server).

Upon successful sign-in, you see the **Server Settings** option in the left-hand side of the screen. Clicking **Server Settings** takes you to the server administration screen. Each option is described there. For more information you can click **Help** in that screen (for example, clicking the **Security** link takes you to the screen shown in Figure 17 on page 34.


Address http://as23/QuickPlace/QuickPlace/Admin.nsf/h_Toc/406AC6104BD821D30525670800167200?OpenDocument

[Go Back]
Instructions
Security
Mail
Existing QuickPlaces
User Directory
Other Options

Security

As an administrator of this server, you can control who can *create new QuickPlaces* a
can enable/disable SSL encryption.

Who can create new QuickPlaces on this server?

[✓ Anyone who can connect to the server.]
Only specific users (or groups) who provide a name and password.

(click Add... to grant create access.)

Who can administer this server? Users and groups in this list can:

- Create new QuickPlaces.
- Decide who *else* can create new QuickPlaces.
- Change server settings, delete QuickPlaces, and create/delete PlaceTypes.

(click Add... to grant admin access.)
admin23

Do you want to use Secure Sockets Layer (SSL) encryption?

SSL is a protocol that securely encrypts data as it travels through the network between web
Note: You must restart the server to make this change effective.

[✓ Do not use SSL]
Use SSL

Figure 17. QuickPlace Server security configuration

When administering the QuickPlace server you can perform the following tasks:

- Secure a QuickPlace Server

  You determine who else can administer the QuickPlace server and who can create QuickPlaces on the server.

- Enable or disable use of SSL

  If the information is sensitive in nature, you can enable Secured Socket Layer access to your QuickPlace server. Enabling SSL encrypts all the network data being passed between the browser client and the QuickPlace server. You can also use the client side certificates, if desired.

- Configure e-mail communication

  If you have installed the QuickPlace server on top of Domino and SMTP is enabled in Domino, then by default QuickPlace uses the Domino services. In standalone mode, you can fill in the appropriate values in the fields

Email Domain and SMTP Server. If you do not have an external SMTP server, then to use the QuickPlace's internal SMTP server, simply leave the SMTP Server field blank.

The Email Domain setting determines whether e-mail can be successfully delivered to Places that are created on this server. The SMTP Server setting determines whether this QuickPlace server can successfully send e-mail notifications to addresses inside and outside your organization.

- Existing QuickPlaces

  You can see a list of all Places on the server, along with their owner and size information. To delete a Place, click the **CleanUp** button.

- Configure User Directory

  If you already have a user directory established in your organization, you can tell the QuickPlace server to use that for name look-up when adding user names to any Place's access control. The QuickPlace server supports Windows NT, Domino, or any LDAP-compliant Directory. That way you can manage the users in a central place. Having a common organizational directory relieves the burden of managing user names in multiple systems.

  When the QuickPlace server is configured to use an external directory, then all user authentication in QuickPlace is performed using that external directory. You can also configure the QuickPlace server not to use any directory, but instead to manage names locally. In this case, user credentials are stored and authenticated with in each Place individually.

- Configure other options: From here you can allow or disallow *Chat* sessions between the users on the QuickPlace server. You can enable or disable execution of Placebots (agents) on the QuickPlace server. This is where you also limit the maximum size of file attachments to any documents on the server.

- QuickPlace PlaceTypes

  PlaceType is like a template. To create a PlaceType, the Place owner first has to authorize that the particular Place can be set up as a PlaceType. With that Place as the base, the QuickPlace server administrator can then create a PlaceType for use by others to create similar Places.

When installing the QuickPlace server, under the advanced section you can specify the TCP/IP address for the QuickPlace HTTP server and also the port. After the installation is done, if you have to change these entries, you can edit the server document in the Domino Directory (names.nsf).

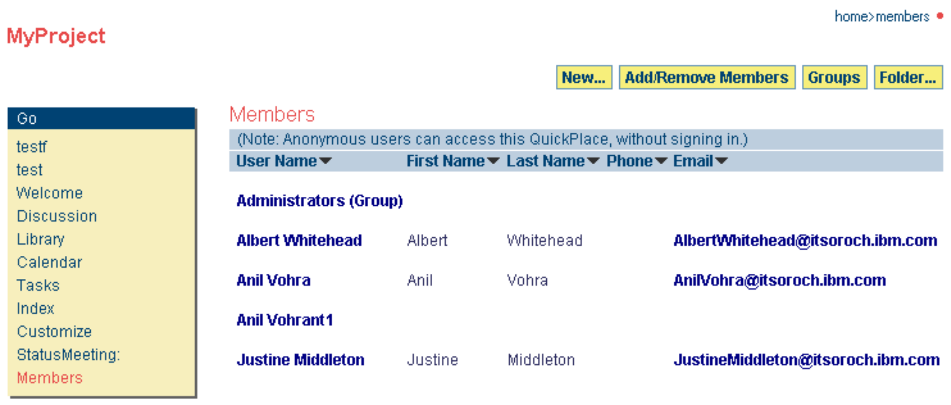
In standalone QuickPlace, you can modify the names.nsf database in the data directory for the port number and for the TCP/IP address. If making a change to the TCP/IP address, remember to change NOTES.INI also.

In the case of a standalone QuickPlace server, end the QuickPlace server before you modify names.nsf because you need to open it as a local database by mapping the drive. In AS/400, if the HTTP server setup job *HTTPSETUP under QDOMINOHT subsystem is running, you should also end that before restarting the QuickPlace, or the changes may not take effect.

3.3.2 Managing the Place instance

The user that creates a Place is initially marked as the manager for that Place. Until or unless you, as an initial manager for the newly created Place, grant access to others, no one other than you can access the Place. Even the QuickPlace server administrator (if you are not that) is unable to see the contents of the Place (however, the server administrator can delete the Place).

For every Place, you can independently specify the readers, authors and managers. Apart from regulating access to the Place, you can further control access to a room and also to a page within the Place. Anonymous access grants rights to everyone. Figure 18 shows an example list of members in our test Place (MyProject); also note the *Groups* and *Folder* buttons.



Members

(Note: Anonymous users can access this QuickPlace, without signing in.)

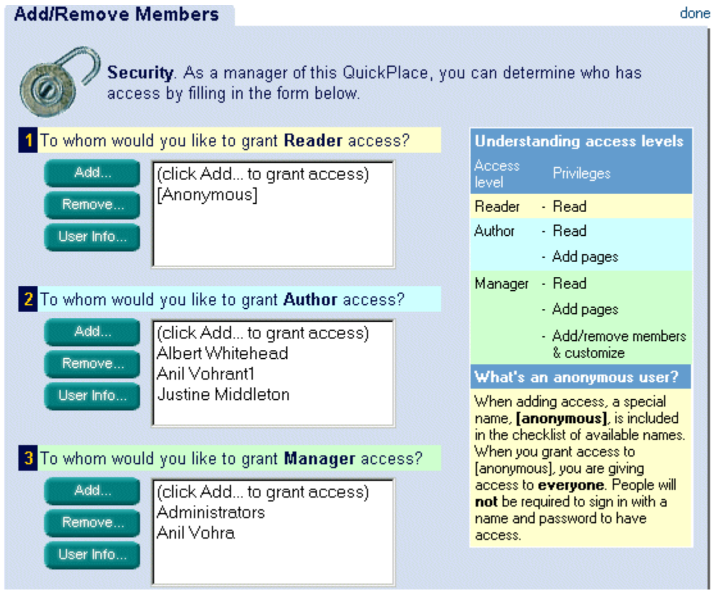
User Name	First Name	Last Name	Phone	Email
Administrators (Group)				
Albert Whitehead	Albert	Whitehead		AlbertWhitehead@itsoroch.ibm.com
Anil Vohra	Anil	Vohra		AnilVohra@itsoroch.ibm.com
Anil Vohrant1				
Justine Middleton	Justine	Middleton		JustineMiddleton@itsoroch.ibm.com

Figure 18. List of members for the test Place (MyProject)

Clicking the **Add/Remove Members** button takes you to the screen shown in Figure 19 on page 37, where you can manage members and their access.

You can create groups within a Place and then use those groups to grant access rights to Rooms within that Place, or notify all members of that group.

Members is the name of a folder. Clicking the **Folder** button lets you change the properties of the current folder. Members is a pre-defined folder, so you may not delete it. User-created folders, however, can be deleted.



Security. As a manager of this QuickPlace, you can determine who has access by filling in the form below.

1 To whom would you like to grant **Reader** access?

(click Add... to grant access)
[Anonymous]

2 To whom would you like to grant **Author** access?

(click Add... to grant access)
Albert Whitehead
Anil Vohrant1
Justine Middleton

3 To whom would you like to grant **Manager** access?

(click Add... to grant access)
Administrators
Anil Vohra

Understanding access levels

Access level	Privileges
Reader	- Read
Author	- Read - Add pages
Manager	- Read - Add pages - Add/remove members & customize

What's an anonymous user?

When adding access, a special name, **[anonymous]**, is included in the checklist of available names. When you grant access to [anonymous], you are giving access to **everyone**. People will **not** be required to sign in with a name and password to have access.

Figure 19. QuickPlace security for the test (MyProject) Place

An entry o*f [Anonymous]* in the Reader access area indicates that anyone can read the information contained in the Place. Two persons have full control (Manager access) over this Place. Apart from the two managers, only one additional person (Author access) can add/edit the information in this Place. Refer to Figure 13 on page 27 for further details.

If your QuickPlace server is configured to use a Domino Directory or LDAP Directory, you can also use groups from that external Directory to control access to the Place. To use groups from the external directory to control the elements inside a Place, you have to first grant that external group some sort of access (reader or author or manager) in the QuickPlace security screen.

Another place you can control access is in page creation. When ready to publish a page, you can use the *Publish As* option to limit the authors and readers of that page. This option also allows you to choose a folder that the page should go in.

You can delete a Place by going into the *Customize* and then *Basics* areas.

Note: Once deleted, a Place cannot be recovered other than from backup media.

Refer to *QuickPlace 2.0 User's Guide*, Chapter 4 "Managing Your QuickPlace" for complete details.

3.3.3 Directory integration

You can set up the QuickPlace server to manage its own directory. The thing to remember in this case is that *every Place maintains its own list of users and their credentials*. So, for example, if John Doe is accessing three different Places on the same QuickPlace server, he will have a separate entry in the corresponding contacts1.nsf of each Place, potentially with a different user ID and a password.

Another option is to use an existing external directory. A Domino Directory or any external LDAP-based directory can be leveraged by QuickPlace.

Note: QuickPlace 2.0 does not support direct access to a LDAP directory via SSL. We do have a workaround for this, albeit a knotty one. See Appendix A, "Accessing LDAP directory via SSL in QuickPlace" on page 347 for more information.

3.3.4 Encryption

The QuickPlace server allows you to encrypt sensitive network data that crosses between browser clients and the server. Depending upon how the QuickPlace is installed, the capabilities can be different.

To establish a secure session, the browser client and the QuickPlace server use certificates. A *certificate* is a confirmation of the identity issued by a certification authority. These can be thought of as "digital IDs". A certificate has two components: a public key and a private key. The public key of the user can be stored in the directory and is easily accessible. The private key is available only to the user and is installed in the client (browser, e-mail, etc.) as a certificate.

How security certificates work

Security certificates work in the following way. If John Doe sends an encrypted and electronically signed e-mail to Jane Doe, then John needs the public key of Jane to *encrypt* the data so that when Jane receives the e-mail, she can *decrypt* with her private key. Jane's private key is known only to her and is installed in her client.

John uses his private key to sign the e-mail that he sends.When Jane receives the e-mail, Jane's client validates John's private key signature, using John's public key to certify that John is who he says he is. In short, the match of public and private key signature combination certifies the user.

There are two types of certificates: personal certificate and Web site certificate.

- A *personal certificate* is a kind of guarantee that you are who you say you are. This information is used when you send personal information over the Internet to a Web site that requires a certificate verifying your identity.
- A *Web site certificate* states that a specific Web site is secure and genuine. It ensures that no other Web site can assume the identity of the original secure site.

The digital signature component of a security certificate is your electronic identity card. The *digital signature* tells the recipient that the information actually came from you and has not been forged or tampered with.

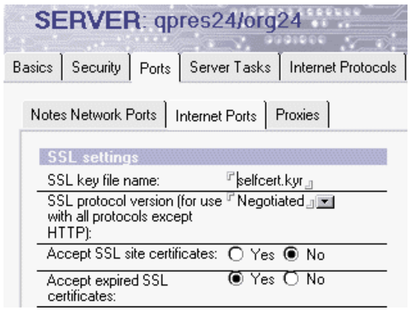
Before you can start sending encrypted or digitally signed information, you must obtain a certificate and set up the client to use it. When you visit a secure Web site (one that starts with https), the site automatically sends you its certificate.

Where to get certificates

Security certificates are issued by independent certification authorities. There are different classes of security certificates, each one providing a different level of credibility. For intranet implementation, a Domino server can be configured as a certificate authority.

When creating a certificate, two files are required: keyring (.kyr), and stashed password (.sth). If running the QuickPlace server in standalone mode, the installation process generates the two certificate-related files as *QuickPlace.kyr* and *QuickPlace.sth.*

When running on top of Domino server, you need to modify the server document and provide the name of the keyring file, as shown in Figure 20 on page 40. You may use Domino as a certificate authority and create the files, or you may request a certificate from a third-party certificate authority. Remember to copy the keyring and stash files to the data directory on the server.


SERVER: qpres24/org24

SSL settings

SSL key file name:	selfcert.kyr
SSL protocol version (for use with all protocols except HTTP):	Negotiated
Accept SSL site certificates:	Yes [✓ No]
Accept expired SSL certificates:	[✓ Yes] No

Figure 20. Place to specify keyring file in the Domino server document

Note: When using a Domino server as a certificate administrator, if you make a request for a certificate using a Notes client that is not on the server console, the certificate files are created *on your client* or *your map network drives*.

For further details, refer to the redbook *Lotus Notes and Domino R5.0 Security Infrastructure Revealed*, SG24-5241, and to the Redpaper *Domino Certification Authority and SSL Certificates*, REDP0046 (available only in softcopy) from the IBM Redbooks Web site:

```
http://ibm.com/redbooks
```

3.3.4.1 SSL

Activating Secured Socket Layer (SSL) communication enables the QuickPlace server to respond to https requests. In the browser you can see the lock at the bottom (on the right-hand side in Microsoft's I.E. and on the left-hand side in Netscape) confirming that the connection is secured (encrypted).

When SSL is enabled, the default communication port is 443 (port 80 is the default for http). This port number can be changed, if desired, during the QuickPlace server installation (only in standalone mode) or by modifying names.nsf after the installation.

When QuickPlace is installed on top of Domino, changing the port to SSL in the server security section of QuickPlace server is not enough. The SSL port setting in the Domino server document also has to be enabled and a valid SSL key file name is required. If you want your QuickPlace server to be available only via https, then you need to disable the TCP/IP port status setting and enable the SSL port status setting in the server document in *names.nsf*. This setting is located under **Ports** and then **Internet-Ports** tab.

Note: If SSL is enabled and a valid key file is not supplied, the HTTP task in the Domino server logs the error. We used the 5.0.4 Domino server release. In some of the previous Domino releases, the HTTP task quits after logging the error.

If you enable the SSL in names.nsf, then configuring the SSL option in the QuickPlace server security screen via the browser (which writes in NOTES.INI) has no impact.

Note: When using I.E., only enable SSL 2.0, SSL3.0 and PCT 1.0--but not TLS 1.0--in the advanced section under **Tools->Internet Options**.

3.3.5 Session-based authentication

When session-based authentication is not enabled then, depending upon the QuickPlace security set-up, the user may be prompted to sign in for every QuickPlace the user visits, even if the log-in credentials are the same. Session-based authentication lets users enter their login ID and password just once.

Session-based authentication can be enabled in the Domino server document. To modify the server document on a standalone QuickPlace server, the *names.nsf* is located in the data directory. You can open it via a Notes client as a local database after shutting down the QuickPlace server. In our case, *names.nsf* was located in the directory c:\lotus\quickplace\data in the Windows NT environment, and in the directory /lotus/quickplace/as23 in the AS/400 environment.

When the QuickPlace server is installed as an overlay, you can open the names.nsf on the Domino server via a Notes client as a server database without shutting down the Domino server. For the changes to take effect, restart the Domino server.

The following two entries need to be added to the NOTES.INI file:

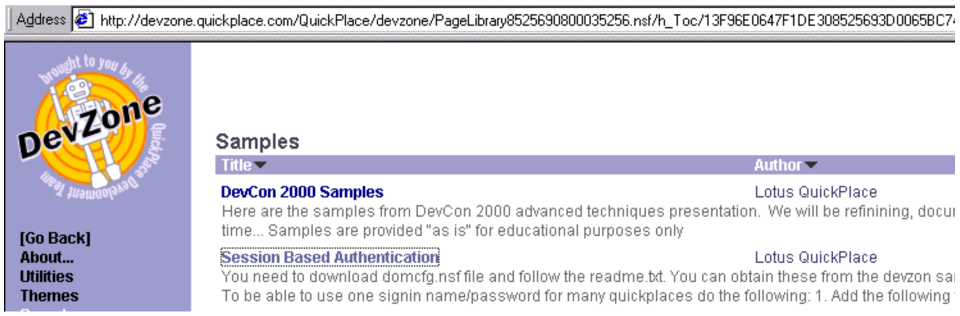
```
h_ScopeUrlInQP=1
NoWebFileSystemACLs=1
```

Keep in mind, though, that this disables any ACLs you may have set for the files or directories via the Domino server.

You must also download *domcfg.nsf* from the QuickPlace DevZone and copy it in the data directory. Standalone QuickPlace server installation does not come with any. If you are running a Domino server and have not modified domcfg.nsf, then just replace the existing one. If you have modified domcfg.nsf, then copy the form *QuickPlaceLoginForm* from the downloaded domcfg.nsf into your copy. You then create a *Mapping a Login Form* entry, in domcfg.nsf, for your QuickPlace server. You may have to use the virtual host setting. You may also modify the existing log-in form. Check the similar entry in the downloaded copy of domcfg.nsf.

The QuickPlaceLoginForm is a modified version of the *$$LoginUserForm* form (the default login form used by Domino during session-based authentication) that is used by the QuickPlace environment to prompt the user for ID and password authentication. The form contains an additional computed text field that appends *?login* at the end of the URL. The tag *?Login* at the end of the URL is used by Domino to prompt the user for credentials when session-based authentication is enabled.

In session-based authentication mode, Domino stores the session ID with the browser client. This session ID (token) enables the server to keep track of the browser client and thus avoids the need to enter user ID and password again. The tag *?Logout* at the end of the URL closes the user session (log-out). The session parameters, like maximum sessions allowed, idle session time-out, etc., can be configured in the server document under the **Internet Protocols->Domino Web Engine** tab.


Address http://devzone.quickplace.com/QuickPlace/devzone/PageLibrary8525690800035256.nsf/h_Toc/13F96E0647F1DE308525693D0065BC7

Samples

Title	Author
DevCon 2000 Samples Here are the samples from DevCon 2000 advanced techniques presentation. We will be refinining, docu time... Samples are provided "as is" for educational purposes only	Lotus QuickPlace
Session Based Authentication You need to download domcfg.nsf file and follow the readme.txt. You can obtain these from the devzon sa To be able to use one signin name/password for many quickplaces do the following: 1. Add the following	Lotus QuickPlace

Figure 21. Session-based authentication information on DevZone

As shown in Figure 21, you can obtain full information from the DevZone URL:

```
http://www.quickplace.com/devzone
```

Look in the **Home : Goodies : Samples** section, or search for a document titled *Session-Based Authentication* .

3.3.6 Single sign-on

When users need to log into many different systems (such as file and print, e-mail, Web applications, etc.), each system may require a user ID and a password. Single sign-on allows the users to be able to enter their log-in credentials once and obtain access to all the authorized network resources.

In a Windows NT environment, if QuickPlace is running on top of Domino, then using Microsoft's IIS as the HTTP stack for Domino allows leveraging the Windows NT security model. The user authentication can be linked so that once the Windows NT and/or IIS authenticates the user, those credentials are passed on to the Domino server (and hence to the QuickPlace server). Single sign-on, together with session-based authentication, can provide users hassle-free access in a distributed environment.

Note: We discovered that this works only for the I.E. browser. Our test with I.E. 5.0. Netscape Navigator was not successful. (We used Navigator 4.7 to test.) Session-based authentication worked with both browsers.

In an AS/400 environment, the single sign-on feature is limited to password synchronization between the Domino client and NetServer. QuickPlace, being a browser-based application, is unable to leverage that. Session-based authentication worked without any issues.

The objective of this section is to briefly mention what capabilities and options exist. From a Windows NT perspective, refer to Chapter 6 in the redbook *Lotus Notes and Domino R5.0 Security Infrastructure Revealed*, SG24-5341, where this topic has been discussed in detail. From an AS/400 point of view, refer to Chapter 12 in the redbook *Lotus Domino for AS/400 R5: Implementation*, SG24-5592.

3.4 Performance

This redbook is dedicated to QuickPlace customization; we did not investigate QuickPlace performance characteristics. However, you can find performance information in the QuickPlace DevZone at:

```
http://www.quickplace.com/devzone
```

Look in the **Home : Technical Docs : Room Index** section or search for *Performance Paper*. At the time of writing, there was performance information for QuickPlace release 2.0, 2.0.5 and 2.0.6.

Table 2, which is an extract from the paper, shows how many users you can have on a QuickPlace server for two different usage scenarios.

Table 2. Max. number of concurrent users from benchmark report in QuickPlace DevZone

Release	Server	Light Publishing	Active Publishing
QP2.0.6	Single Intel 450Mhz CPU, 512 Mbyte RAM	360	336
QP2.0.6	Sun Ultra60, dual 450 Mhz, 512 Mbyte RAM	510	448
QP2.0.6	Dual Intel 733Mhz CPU, 523 Mbyte RAM	1600	1400
- - -			
QP2.0.5	Single Intel 450Mhz CPU, 512 Mbyte RAM	300	280
QP2.0.5	Sun Ultra60, dual 450 Mhz, 512 Mbyte RAM	420	392
- - -			
QP2.0	Single Intel 450Mhz CPU, 512 Mbyte RAM	140	126

The two scenarios are defined as follows:

- Light Publishing Scenario

  Users read different page types (simple text page, imported page, calendar page) and click table of content entries. Each user creates roughly one page request every minute. Also, some users publish simple text pages once every 10 minutes. 80% of the users perform reading tasks, while 20% of the users perform publishing tasks.

- Active Publishing Scenario

  This includes all activities from the Light Publishing Scenario plus publishing imported MS Word documents and creating calendar pages. Each user publishes simple text pages, calendar pages or MS Word imported pages once every ten minutes. 60% of the users perform reading tasks, while 40% of the users perform publishing tasks.

The paper also includes the following interesting findings:

- Number of Places

  The number of Places on the server has not been found to affect server performance.

- Size of QuickPlace views

  Initial performance tests have shown that QuickPlace server processing is sensitive to the number of documents in a view or folder. If your users produce QuickPlaces with views containing more than 100 documents, it's recommended that you reduce the number of total users by a factor of 2.

- Domino overlay

  If the QuickPlace server is an overlay on a Domino server, then the load on the Domino server needs to be taken into account. Because QuickPlace requests and Domino server requests (Web-mail, etc.) can be treated as distinct and separate, the number of total users using the Domino server should be subtracted from the number of total users you calculate for QuickPlace access to get the load that the QuickPlace server component can sustain.

Performance testing of the QuickPlace server is ongoing, so be sure to check the QuickPlace DevZone for full and latest information.

3.5 Summary

In this chapter we covered the directory structure of the QuickPlace server. We examined installing QuickPlace in standalone or overlay mode, and we also described how QuickPlace handles certifiers and controls security.

Chapter 4. Creating Themes

QuickPlace Themes allows you to add your unique graphic identity and look and feel to your application while the QuickPlace server handles all the functionality.

In this chapter we first explain what a Theme is and which elements it contains, including the stylesheet file. We then walk through the steps to get an existing Theme to modify and how to upload it once it is modified. We discuss the process we used when creating the Theme for our sample company *Millennia Consulting*. We will include reference information about the stylesheet class names QuickPlace uses and their default values. We will also include reference information for all the different QuickPlace layout components, and give examples of how they can be used. Finally, you will see how we added extra functionality to our Millennia Theme when we described the Favorites page that was added to the Millennia Theme.

4.0.1 What is a Theme

A QuickPlace Theme controls the look and the layout of a QuickPlace: its fonts and background colors; how an element looks when it is selected; where the navigational controls appear; and so on. When you create a QuickPlace, you can select a Theme by choosing from a gallery of predefined Themes. You can also modify an existing Theme or create a new Theme.

Using a custom Theme, you can give your QuickPlace a strong brand identity, or design it to look like other corporate sites, or supply additional functionality as well as a unique look.

4.0.1.1 Anatomy of a QuickPlace Theme

Each Theme is composed of a group of HTML files, a CSS file that defines the appearance of specific QuickPlace components, a Gallery image, and images used in the layouts. For example, the layout for a page differs from the layout of a folder, but they will probably share some style elements as part of a common Theme. As listed in Table 3, a QuickPlace Theme is composed of the following stylesheet, layouts, gallery image, and images:

Table 3. Anatomy of a QuickPlace Theme

Layout File	Type	Purpose
Stylesheet	.css	Defines styles such as fonts and colors for all layouts.

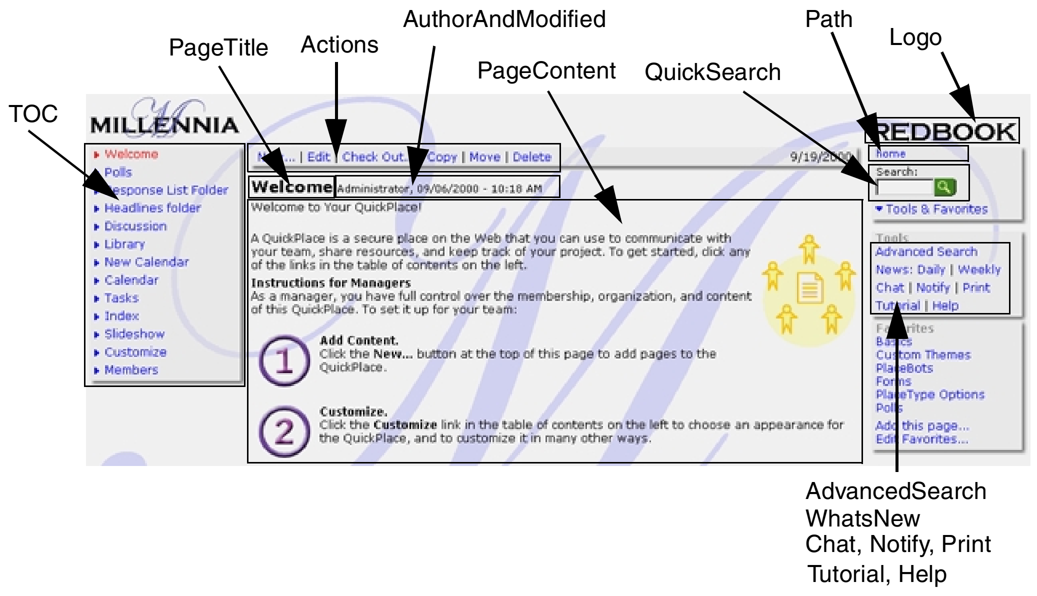
Layout File	Type	Purpose
Page	.htm	Defines the appearance of a page being read
Page editing	.htm	Defines the appearance of a page being edited
List folder	.htm	Defines the appearance of a List or Response folder
Headlines folder	.htm	Defines the appearance of a Headlines folder
Slideshow folder	.htm	Defines the appearance of a Slideshow folder
Gallery image	.jpg or .gif	Image to represent the Theme
Images	.jpg or .gif	Images used in the layout files

Note: The default Themes can be found in the *resources.nsf* database that comes with QuickPlace.

Note: To upload customized Themes, you must use Internet Explorer.

4.0.1.2 Custom Themes and HTML

Themes are implemented using the QuickPlace layout architecture and are defined using HTML. To customize a Theme, edit the HTML in your favorite HTML editor and upload the modified files. QuickPlace provides a set of custom HTML tags that you use to define the elements in each layout. In Figure 22 on page 49 you can see examples of the elements (also called layout components) we used in our Millennia Theme.



Welcome Administrator, 09/06/2000 - 10:18 AM

Welcome to Your QuickPlace!

A QuickPlace is a secure place on the Web that you can use to communicate with your team, share resources, and keep track of your project. To get started, click any of the links in the table of contents on the left.

Instructions for Managers
As a manager, you have full control over the membership, organization, and content of this QuickPlace. To set it up for your team:

1. **Add Content.** Click the **New...** button at the top of this page to add pages to the QuickPlace.
2. **Customize.** Click the **Customize** link in the table of contents on the left to choose an appearance for the QuickPlace, and to customize it in many other ways.

Figure 22. Some Millennia Layout components

When you customize a Theme, you can take advantage of all of the power of HTML, DHTML, Javascript and XML to add functionality to a QuickPlace. Here are some ways you might enhance a QuickPlace using custom Themes:

- Apply the corporate brand identity to a QuickPlace or create a custom graphic identity for a collaborative application.
- Integrate a QuickPlace seamlessly, as a collaborative component within a larger corporate Web site.
- Provide hardcoded links from a QuickPlace to other Web sites such as corporate Web sites, eCommerce sites, or to customer support services.
- Make new features available by embedding ActiveX controls or Java applets in the custom Theme.
- Use JavaScript, DHTML and so on to program dynamic effects into the custom Theme (like the Toolsbox in our Millennia example).

4.0.1.3 Custom Themes and JavaScript

JavaScript is an integral part of most advanced Themes, but there are some of things to watch out for. These issues are described with in the following section.

When you upload a layout file, everything between the first <HTML> and the end of the <BODY> tag is removed, including the <HEAD> tag. This is replaced by QuickPlace's own header. For this reason, it is important to specify your <SCRIPT> tags in the body of the document.

When using the JavaScript "document.write" method to write out HTML content in a QuickPlace element, such as an imported page, layout file and so on, it is normal to write to the PageBody field of a document. Note this field is called "PageBody", not "PageContent" which is a QuickPlaceSkinComponent.

Tip: Since the <body> tag is ignored, it makes it impossible to put background images or colors in the <body> tag of the html page. You can get around this by making a class selector in the Style Sheet called *.h-page-bg* and putting the image and/or color in there; see Appendix 4.2.1.1, "What a Style Sheet looks like" on page 59. This is the class that QuickPlace uses for the page background.

Example:

```
.h-page-bg {
   background-image: url(bgM.gif);
   background-repeat: no-repeat;
   background-color: White;
   margin-left: 5px;
   margin-top: 5px;
   margin-right: 5px;
}
```

4.0.1.4 Who should customize a Theme?

You can modify an existing Theme with minimal HTML development skills. For example, if you want to move the QuickSearch component, you only need to find the following tag in your layout file:

```
<QuickPlaceSkinComponent name=QuickSearch>
```

It will most often be placed within a HTML table. With a little knowledge of HTML, you can put the tag in a table specification so it appears where you want it on the page.

To create a custom Theme from scratch, however, requires more advanced expertise with good knowledge of HTML, cascaded style sheets (CSS), and Web development.

4.0.1.5 Custom Themes and PlaceTypes

If you create a customized Theme, you reuse your Theme as part of a template from which you can build similar QuickPlace applications. To do this, save the QuickPlace containing the custom Theme as a PlaceType, which you can then use for creating new QuickPlace applications.

For example, you create a Theme for a company called Millennia. Then you create a new custom Theme, and apply it to the Millennia QuickPlace. By specifying that the Millennia QuickPlace is a PlaceType (or template), you can make new QuickPlaces that look exactly like your original.

4.1 Millennia Theme: General overview

Our client, Millennia wanted to maintain their corporate Web site look and feel but add some new 3-D effects to the layout. Using an existing Web site is a good starting point for the Theme since we have HTML files to work with. Millennia wanted to be able to hide the Tools: chat, notify and so on, when the user wanted to, and also wanted to be able to save user-specific *Favorites*. This was done using JavaScript, DHTML, and cookies.

4.1.1 Creating the Millennia Theme

Creating a Theme from scratch is a process that takes time. It needs a lot of consideration about where to place different sections and Layout Components. In this section we describe the steps that we went through to create the Millennia Theme. The files making up the Theme are available for download from the IBM Redbooks Web site. See Appendix L, “Additional Web material” on page 425 for instructions on how to get the sample files.

1. Start with a printed copy of the client’s Web site (or a Web site you like the design of) and blank printed screen shots of a browser window. With the design next to you, start to draw the different sections the site should contain by hand on the blank pages.

   Keep in mind the different components that QuickPlace uses which are not common on regular Web Pages, like *Revision* (see Appendix 4.3.2.17, “Revision” on page 98) and *Chat* (see Appendix 4.3.2.4, “Chat” on page 83). HTML designers often start to code before considering *all* the elements they need to implement on the pages, so be aware that you also have to consider that there are five different Layouts used in QuickPlace: Pages in read mode, pages in edit mode and three different folder layouts.

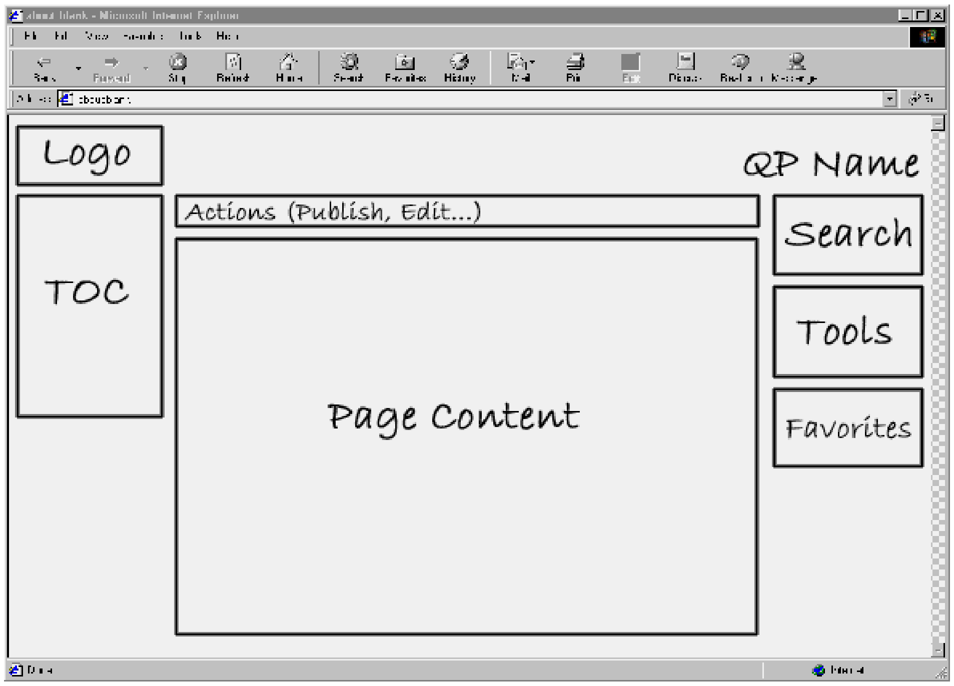


Figure 23. Browser window with Page layout proposal

2. Prior to meeting with the client, let the graphic department make up some quick *examples of the design in HTML* (or do it yourself). Making the examples in HTML instead of a layout program minimizes the risk of showing the client something they later cannot get because of limitations in HTML.

Tip: An easy way of *stealing* Web pages, Style Sheet, External Java Scripts, and images is to go to the Web site with Internet Explorer 5.x and select **Save As...** from the File menu. This puts the HTML file, together with all the external files, on your hard disk.

Instead of putting the QuickPlace Layout Component tags in the HTML, you can “fake” them by actually hardcoding examples of them in there. This gives you the freedom of taking the HTML file with you and showing it on any computer. Otherwise, you must have a QuickPlace Server installed and running on the computer.

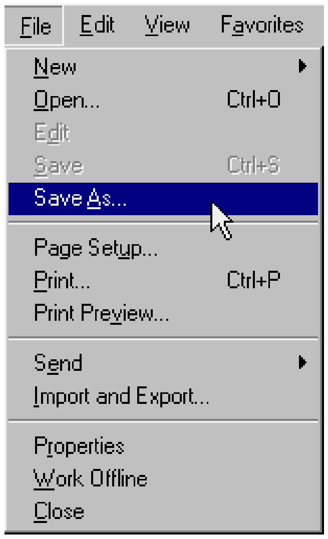


Figure 24. Saving the files from an existing Web site

3. When you meet with the client, discuss the overall layout, with emphasis on the different Layout Components you need in your Place and where the client thinks they should be implemented on the layouts; the client may not want some of the Layout Components on some or all of the layouts.

4. Ask the client if there are any special needs they have that require coding of PlaceBots or design of new forms. If forms need to be made, make sure you know what the requirements are.

5. Determine if there are any other QuickPlace-specific design elements that the client needs, such as rooms, folders, or pages. If there are, make sure you get all information about them, such as who should have access to rooms, which form to use in folders, and so on.

6. Start coding the Style Sheet. We recommend using a Style Sheet editor. This makes it easier to see what the selectors you change look like. Use the existing Style Sheet from a Web site, together with the QuickPlace generated Style Sheet (see Appendix 4.2.2, “Guide to selectors in the standard QuickPlace Style Sheet” on page 62).

   You can use your own *Selectors* (see Appendix 4.2, “Cascading Style Sheets” on page 58) to make the QuickPlace look the way you want, but we recommend using QuickPlace Selector names because the QuickPlace Layout Components in some cases use predefined selector names. Just by changing the colors, background images, and fonts in QuickPlaces Selectors can alter the look of your place considerably.

7. Now it's time to create the HTML layouts. This is where the work comes together. Based on all the information from your client, you can create the layouts. Use any HTML editor that you like. Some people like coding in the Note Pad, but this makes the use of tabulators harder. Replace hardcoded text with Layout Components. For example, in the HTML file there are links to the some other pages that are reachable from this page. These hardcoded links should be replaced by the TOC Layout tag (see Appendix 4.3.2.19, "TOC" on page 100).

Example of old way

```
<font face=arial size=1>
<a href=home.htm><b>Home</b></a>
<p>
<a href=whoarewe.htm>Who are we?</a>
<p>
<a href=help.htm>Help</a>
</font>
```

Example of new way
Style Sheet:

```
.h-toc-text{
   font-family: arial;
   font-size: 9pt;
}
.h-tocSelected-text{
   font-family: arial;
   font-size: 9pt;
   font-weight: bold;
}
```

HTML layout file:

```
<QuickPlaceSkinComponent
   name=TOC
   format={<Item class=h-toc-text>}
   selectedFormat={<Item class=h-tocSelected-text>}
   delimiter={<p>}
   emptyFormat={}
   replaceString={}
>
```

Although this seems like more to write, consider this: if you want to make changes to the font, color or font weight, by using the new way, you only have do to this in the Style Sheet and the changes will be carried out.

4.1.2 Customizing Themes Quick Start

You can download an existing Theme, modify it, and upload it as a new Theme. In this section we describe how to download and upload a Theme.

4.1.2.1 Downloading a Theme

An easy way to start customizing QuickPlace Themes is to tell QuickPlace you want to create a new Theme, then ask QuickPlace to generate the files for your new Theme. (At first, your new theme will look just like the default theme.) You can then make changes to the default Theme (this is a good way of learning how to customize a Theme):

1. Go into a QuickPlace in which you have Manager access.
2. Enter the **Customize** area.
3. Click **Custom Themes** near the bottom of the page.
4. Click **New Theme**.
5. Enter the title of your new theme in section 1. We called ours: `MyTheme`.
6. (Optional) Enter a description of your theme in section 2. We entered as a description: `A variation of the Default Theme`.

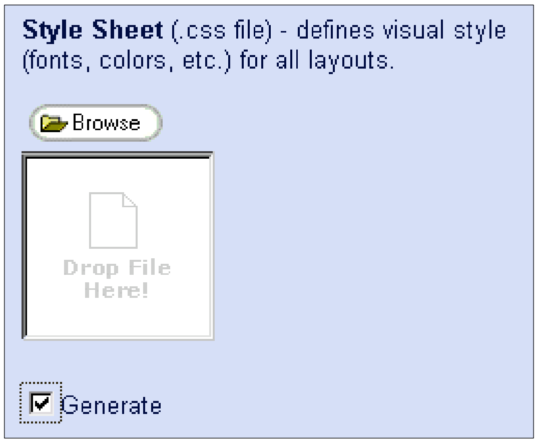

Style Sheet (.css file) - defines visual style (fonts, colors, etc.) for all layouts.

Figure 25. Getting QuickPlace to generate a Style Sheet file

7. Select the **Generate** check box for the Style Sheet and all Layout files. (This will be turned on for most of them, by default.)
8. Click **Next** to save the Theme.
9. Check that your new Theme is visible in the list of Themes. It may be the only one. Now, the Theme has been created, but not implemented.

10. To implement your new Theme, click **Customize** in the Table Of Contents, even though it already appears to be selected.
11. Click **Decorate** with the symbol of the painter's palette.
12. Click **Choose a theme** (1)

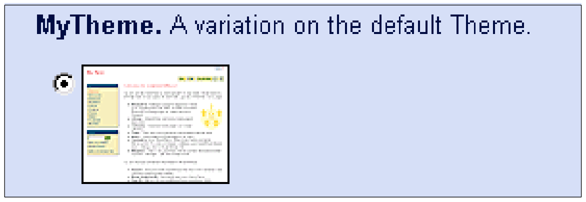

MyTheme. A variation on the default Theme.

Figure 26. Selecting the Theme in the Customize -> Decorate -> Choose a Theme page

13. Scroll down to the bottom of the theme and select the newly created Theme. In our case, this was **MyTheme**.
14. Click **Next**. Now the Theme will be implemented; however, you will not see any difference to the default Theme. The reason for doing this is that now we have implemented a Theme that we can manipulate.
15. Click **Customize** again**.**
16. Click **Custom Themes**
17. Click the name of your theme.
18. Look into the first upload control (also called a *bucke*t), and make sure that it has a file called **StyleSheet.css** in it.
19. Click **Generate** to remove the check box selection.
20. Make sure that you have a folder visible at the side of your Browser window.

21. Drag first the **StyleSheet** and then the **Page** files across into the folder on your hard disk.

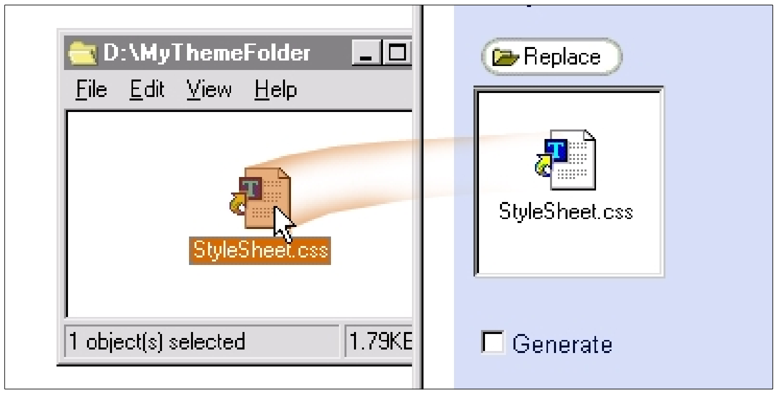


Figure 27. Copying a file from the StyleSheet bucket into a folder.

22. Open a text editor.
23. Go to the file menu and open the file via the Open option.
24. You can now make changes to the files to begin the customization of your QuickPlace.

If you have made changes to these new files and you want to upload them, refer to the next section on uploading Themes.

4.1.2.2 Uploading a Theme

The following section describes how to upload a Theme into QuickPlace. It assumes that you have successfully created a Theme and now want to put it into your QuickPlace.

1. Save the file you have created as a StyleSheet (CSS) file or Layout file (HTM).
2. If you are in the Edit Theme Page of your QuickPlace already, go to step 5 of this section.
3. Go to your QuickPlace, click **Customize**, and scroll down and click **Custom Themes**.
4. If you want to edit an existing Theme, click **New Theme;** otherwise, click the name of the Theme you want to edit, in our case **MyTheme**.
5. Make sure that the **Generate** tickmarks have been turned off in the StyleSheet and Page Layout check boxes.
6. Click the **Replace** Button above the bucket.

7. Find the CSS file that you have created or modified, and click **Open** in the dialog box.
8. The new file has now been uploaded.
9. Repeat steps 6 and 7 for the Page Layout file.
10. Click the **Next** button at the top or bottom of the Edit page to upload your new files. If you look carefully, you will see all your images' names appearing as they are being uploaded by QuickPlace.
11. Your new Theme has been created and uploaded. If the changes do not appear, read the following steps on Implementing a Theme.

Note: If your images do not appear as you would expect, make sure that the image paths are correct. For example, we created our layout files in the folder called MyThemeFolder. All the images were in the same folder as the Page.htm file, so our TML image tags looked like this:

```
<img src=transparent.gif>
```

In this image there are no directories specified before the image name (transparent.gif). If the images we wanted to include were in a separate folder, the directory structure would have to reflect that.

4.1.2.3 Using a new Theme in QuickPlace

In order to tell QuickPlace to use the Theme you created or modified, you need to select it via the Customize -> Decorate page, as follows:

1. Go into a QuickPlace in which you have Manager access.
2. To implement your new Theme, click **Customize** in the Table Of Contents, (even if it already appears to be selected).
3. Click **Decorate** with the symbol of the painter's palette.
4. Click **Choose a theme** (1)
5. Scroll down to the bottom of the Theme and select your Theme.
6. Click **Next**.

Now the Theme will be one that is used in your Place.

4.2 Cascading Style Sheets

The Style Sheet is a very important element of a QuickPlace Theme. In the following sections we give a general introduction to Style Sheets, discuss specific QuickPlace considerations, and list the StyleSheet selectors used by QuickPlace.

4.2.1 What Style Sheets are

If you’ve ever created a Web site, you’ve most likely used an assortment of <font> and other tags to control the look of the pages. This ties the look of your Web site to whatever tags you used, making it a tedious process to update or modify your site's design, because if you later decide to change the fonts or colors used in your site, you have to *edit every page* to do this.

With QuickPlace, however, instead of defining the site's design in each and every page, you can use a Style Sheet to control the overall layout of your site. Then if you want to change how your site looks, you simply modify the Style Sheet.

4.2.1.1 What a Style Sheet looks like

A Style Sheet is made up of rules that look something like this:

```
H3 {
    font-family: Arial;
    font-style: italic;
    color: green
   }
```

Each rule begins with a selector. A *selector* is normally one or more HTML elements (tags) or a class, and the declarations inside the rule describe how those elements should appear. A declaration is simply a CSS property followed by a value. For example, the declaration " font-family: geneva; " is composed of the property " font-family " followed by the value " geneva ".

```
body {
   font-family: geneva, arial, sans-serif;
   font-size: 10pt;
   color: #333333;
}
```

You can also use classes as selectors, which are not tied to specific HTML elements. This is used extensively in QuickPlace. For example, consider this rule:

```
.h-folderBanner-text {
    font-family: arial;
    font-style: italic;
    color: green
    }
```

The declaration block is the same as that in the previous example, but instead of using body as the selector, we're using the class .h-folderBanner-text. Note that .h-folderBanner-text doesn't mean anything special - you can use anything as a class name, provided that it *starts with a period* and is composed of a *single word* (note that *spaces and underscores are not allowed*).

Tip: A good practice is to name classes according to their function, rather than their appearance.

To apply a class to an HTML tag, use the class attribute. For example, to apply the above style to an <td> tag, you'd use:

```
<td class=h-folderBanner-text>this is some text</td>
```

Note: Period before the class name is not included.

4.2.1.2 How to use Style Sheets in the layouts

In QuickPlace, upload the Style Sheet in the Customize -> CSS StyleSheet bucket (Upload Control).

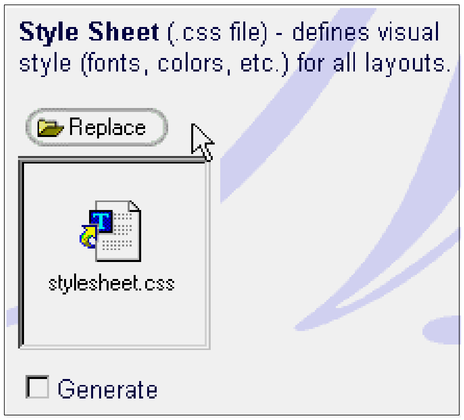

Style Sheet (.css file) - defines visual style (fonts, colors, etc.) for all layouts.
Replace
stylesheet.css
Generate

Figure 28. The StyleSheet bucket (Upload Control)

Tip: When creating a Layout file, we add a <STYLE> tag into the <HEAD> tag of the layout document, referencing the CSS StyleSheet.css page. This tag is removed when the Layout file is uploaded to a QuickPlace, but replaced by the StyleSheet.css file that you upload in the Customize -> CSS StyleSheet Bucket (Upload Control). Doing this allows us to preview our HTML page before we convert it to a Layout file. In our Layout file, that line looked like this:

```
<link rel=stylesheet" type="text/css" href="stylesheet.css">
```

4.2.1.3 Grouping

In order to decrease repeating statements within Style Sheets, grouping of selectors and declarations is allowed:

```
.h-toc-text, .h-tocSelected-text {
   font-family: geneva, arial, helvetica, sans-serif;
   font-size: 9pt;
}
```

These two classes, *.h-toc-text* and *.h-tocSelected-text*, look the same when used in our layouts. They have the same font-family and same font-size.

4.2.1.4 Inheritance

Virtually all selectors which are nested within selectors will inherit the property values assigned to the outer selector unless otherwise modified. For example, a color defined for the BODY will also be applied to text in a paragraph.

There are some cases where the inner selector does not inherit the parent selector's values, but these should stand out logically. For example, the margin-top property is not inherited; intuitively, a paragraph would not have the same top margin as the document body.

4.2.1.5 Comments

Comments are denoted within Style Sheets with the same conventions that are used in C programming. A sample CSS comment would be in the format:

```
/* This is a comment */
```

4.2.1.6 Anchor Pseudo-classes

Pseudo-classes can be assigned to the *<a>* element to display links, visited links, active links and hover links differently.

```
a.h-toc-text:hover, a.h-tocSelected-text:hover {
   color: #e74d4a;
}
```

4.2.1.7 Cascading order

When multiple Style Sheets are used, or when one Style Sheet with a selector is named more than one time, the browser needs to know which selector to use.

In these situations, there must be rules as to which Style Sheet's rule will win out. The following characteristics will determine the outcome of contradictory Style Sheets.

4.2.1.8 The ! important statement

A style that is designated as important will win out over contradictory styles of otherwise equal weight. Following is a sample use of the ! important statement:

```
.h-fieldSmallEdit-text {
   font-family: geneva, "ms sans serif", sans-serif ! important;
   font-size: 9pt ! important;
}
```

The ! important statement should only be used when the selector in the QuickPlace-generated Style Sheet is using ! important and you want to change that selector.

4.2.1.9 Order of Ssecification

To make it easy, when two rules have the same weight, the last rule specified wins. This is important to know since QuickPlace generates the basic selectors described below in the Style Sheet and then append your selectors to them. This means that if you create a class in your Style Sheet with the same name as in the QuickPlace-generated Style Sheet, your declarations take over.

```
/* First occurrence */
.h-toc-text {
   font-family: arial, helvetica, sans-serif;
   font-size: 9pt;
}

/* Second occurrence */
/* This is what .h-toc-text will look like */
.h-toc-text, .h-tocSelected-text {
   font-family: verdana, arial, helvetica, sans-serif;
   font-size: 10pt;
}
```

In the example above you can see that the class *.h-toc-text* is named twice, first by itself and then grouped together with *.h-tocSelected-text*. *.h-toc-text* will have the declarations of the second occurrence.

4.2.2 Guide to selectors in the standard QuickPlace Style Sheet

Notes:

1. Selectors in **bold** are the main selectors that you need to change to broadly restyle a Theme. The other selectors provide additional control over specific types of content.

2. A standard default stylesheet is always output with any Theme (column 4), so that you only need to specify the selectors that you wish to change. Undefined properties will fall back to those defined in the default stylesheet.
3. Imported content is styled as follows:
   - The styling of imported content that has explicit HTML styling (such as **font** tags), or an existing Style Sheet, is preserved when the page is imported.
   - Unstyled content inherits the styles defined by the current Theme.
4. *Debugging tip:* To view the current Style Sheet on any active QuickPlace page, exactly as it is delivered to the browser, follow these steps:
   a. Right-click in the page for which you want to see the Style Sheet and do "View Source".
   b. Search in the HTML source for the first occurrence of **<LINK**, which will locate the link tag of the prevailing Style Sheet.
   c. Select and copy the relative URL of the Style Sheet (the text inside quotes following "href=" inside the link tag).
   d. Construct the absolute URL by prefixing the relative URL with the server domain name (e.g., **www.quickplace.com**).
   e. Enter the resulting URL into the browser address/location bar to view the Style Sheet:

```
http://www.quickplace.com/QuickPlace/redbook/Main.nsf/$skip/$defaultview/9
EEB7D0C2C9DF92405256820007751D3?OpenDocument&Form=h_StyleSheet&CacheResult
s&TimeStamp=8525695700626C3F&LoginName=name%2FOU=password%2FOU=QP%2FO=quic
kplacename
```

Note: Although the automatically created Style Sheet (shown in column 4 of the following table) sometimes blocks several Style Sheet classes together, you can tweak your classes independently.

Table 4. Reference Guide to Style Sheet selectors in QuickPlace

CSS selector	Description	Notes	Standard styles in QuickPlace
Tag styles			

body, td	Default text style	Specify both tags to set the default text style	body, td {
			font-family: geneva, arial, sans-serif;font-size: 10pt;color: #333333;
			}

CSS selector	Description	Notes	Standard styles in QuickPlace

a	Anchor style	See also several other more specific anchor styles, below	a {
			color: navy;
			font-weight: bold;
			text-decoration: none;
			}

a:hover	Default style of anchors when mouse is over the anchor	IE only	a:hover {
			color: red;
			text-decoration: underline;
			}

h1, h2, etc	Regular styles		h1 { font-size: 13pt; font-weight: bold; }
			h2 { font-size: 12pt; font-weight: bold; }
			h3 { font-size: 12pt; }
			h4 { font-weight: bold; }
			h5 { text-transform: uppercase; }
			h6 { font-style: italic; }

form	Default style of forms	The **margin-bottom** property is set to **0px** by default to remove unwanted whitespace from the bottom of all form	form {
			margin-bottom: 0px;
			margin-top: 0px;
			padding-top: 0px;
			padding-bottom: 0px;
			}

Page background			

.h-page-bg	Page background	Class assigned to body tag of all pages. For IE only, the **margin** properties can be set to control the page margin	.h-page-bg {
			background-color: White;
			}

Folders, What's New, Search Results, Tasks (list view)			

CSS selector	Description	Notes	Standard styles in QuickPlace

.h-folderBanner-bg	Background of folder banner	This style is used for the banner that displays column titles, as well as other banners in What's New, Search Results, etc.	.h-folderBanner-bg {
			background-color: #bcd;
			padding-left: 1px;
			padding-right: 1px;
			}

.h-folderBanner-text	Text in folder banner	-- ditto --	.h-folderBanner-text {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 9pt;
			font-weight: normal;
			color: #004573;
			}

a.h-folderBanner-text	Anchors in folder banner	-- ditto --	a.h-folderBanner-text {
			font-weight: bold;
			text-decoration: none;
			}

.h-folderBannerSelected-text	Text of selected ("current") item in folder banner	-- ditto --	.h-folderBannerSelected-text {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 9pt;
			font-weight: bold;
			color: black;
			text-decoration: underline;
			}

a.h-folderBannerSelected-text	Selected anchor in folder banner	-- ditto --	a.h-folderBannerSelected-text {
			font-weight: bold;
			text-decoration: underline;
			}

.h-folderItem-bg	Background of items listed in folder	-- ditto --	.h-folderItem-bg {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 4pt;
			}

CSS selector	Description	Notes	Standard styles in QuickPlace

.h-folderItem-text	Text of items listed in folder	-- ditto --	.h-folderItem-text {
			color: #666;
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 9pt;
			padding-left: 1px;
			padding-right: 1px;
			}

a.h-folderItem-text	Anchor listed in folder	-- ditto -	a.h-folderItem-text {
			font-weight: bold;
			text-decoration: none;
			color: navy;
			padding-left: 0px;
			}

.h-folderCompact-text	Compact text of item listed in folder	-- ditto --	.h-folderCompact-text {
			color: #336;
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 9pt;
			font-weight: normal;
			padding-left: 1px;
			}

.h-folderAbstract-text	Abstract text of item listed in folder	-- ditto --	.h-folderAbstract-text {
			color: #666;
			font-weight: normal;
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 9pt;
			padding-left: 1px;
			}

.h-folderBar-bg	Background of bar to left of a thread	-- ditto --	.h-folderBar-bg {
			background-color: rgb(91,135,165);
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 0pt;
			padding-left: 1px;
			padding-right: 1px;
			}

CSS selector	Description	Notes	Standard styles in QuickPlace

.h-folder-dl	Indentation of responses in response folder	By default, the margin-bottom property is set to 0px to remove unwanted whitespace below indented items in response folders	.h-folder-dl {
			margin-bottom: 0px; /* Space after indented rows */
			}

.h-folderInterspace-bg	Background color of vertical space between responses		.h-folderInterspace-bg { }

.h-folderInterspace-text	Height of vertical space between responses	Use **font-size** to set the height	.h-folderInterspace-text {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 4px; /* Space between responses */
			}

.h-folderSpace-text	Height of vertical space between threads	Use **font-size** to set the height	.h-folderSpace-text {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 9pt; /* Space between threads */
			padding: 0px;
			}

What's New			

.h-whatsNewBanner-bg	Background of outer box in right column of What's New		.h-whatsNewBanner-bg {
			background-color: #f7efbd;
			padding-left: 1px;
			padding-right: 1px;
			}

CSS selector	Description	Notes	Standard styles in QuickPlace

.h-whatsNewBanner-text	Text of outer box in right column of What’s New		.h-whatsNewBanner-text {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 10pt;
			line-height: 10pt;
			color: #004573;
			padding-left: 1px;
			padding-right: 1px;
			}

.h-whatsNewBox-bg	Background of inner box in right column of What’s New		.h-whatsNewBox-bg {
			background-color: #ffe;
			}

.h-whatsNewBullet-text	Bullet to left of items listed in What’s New		.h-whatsNewBullet-text {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 14pt;
			line-height: 10pt;
			color: #e74d4a;
			text-indent: 0.25em;
			}

QuickBrowse “remote control” window			

.h-quickBrowseTitle-text	Title displayed in QuickBrowse window		.h-quickBrowseTitle-text {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 10pt;
			font-weight: bold;
			color: #004573;
			padding-left: 3px;
			padding-right: 3px;
			}

.h-quickBrowseBullet-text	Bullet to left of items listed in QuickBrowse		.h-quickBrowseBullet-text {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 14pt;
			line-height: 9pt;
			color: #e74d4a;
			text-indent: 0.25em;
			}

CSS selector	Description	Notes	Standard styles in QuickPlace

.h-quickBrowseItem-text	Text listed in QuickBrowse		.h-quickBrowseItem-text {
			font-family: geneva, "ms sans serif", sans-serif;
			font-size: 9pt;
			line-height: 9pt;
			color: #666;
			padding-left: 1px;
			text-indent: 0px;
			margin-left: 0px;
			}

a.h-quickBrowseItem-text	Anchor listed in QuickBrowse		a.h-quickBrowseItem-text {
			color: navy;
			font-weight: bold;
			line-height: 9pt;
			text-decoration: none;
			}

.h-quickBrowseNav-text	Navigation link displayed in QuickBrowse		.h-quickBrowseNav-text {
			font-family: geneva, "ms sans serif", sans-serif;
			font-size: 9pt;
			font-weight: normal;
			color: #004573;
			}

Tasks (timeline view)			

.h-tasksBannerNow-textbg	Highlighted current date in Tasks banner		.h-tasksBannerNow-textbg {
			color: White;
			font-weight: bold;
			background-color: #e74d4a;
			}

.h-tasksItem-bg	Background of items listed in Tasks		.h-tasksItem-bg {
			background-color: #f0f7ff;
			}

.h-tasksItemTimeline-bg	Highlighted period of a task		.h-tasksItemTimeline-bg {
			background-color: #d66;
			}

.h-tasksItemMilestone-bg	Highlighted period of a milestone		.h-tasksItemMilestone-bg {
			background-color: #0a0;
			}

Calendar			

CSS selector	Description	Notes	Standard styles in QuickPlace

.h-calendarLabel-text	Date label		.h-calendarLabel-text {
			color: #004573;
			}

.h-calendarLabelSelected-text	Date label (today's date)		.h-calendarLabelSelected-text {
			color: red;
			}

.h-calendarItemOther-bg	Background of day not in current month		.h-calendarItemOther-bg {
			background-color: #cde;
			}

.h-calendarItemToday-bg	Background of today's date		.h-calendarItemToday-bg {
			background-color: #ffe;
			}

Text and fields in Page layout			

.h-field-text, .h-field-text td	Style of the text value of a field	Use this exact selector, as shown, to style field text distinctly from regular page content	.h-field-text, .h-field-text td {
			}

.h-pageSmall-text,
.h-fieldSmall-text
	"Smallprint" page text and "Smallprint" text content of fields		.h-pageSmall-text,
			.h-fieldSmall-text {
			font-family: geneva, "ms sans serif", sans-serif ! important;
			font-size: 9pt ! important;
			}

.h-fieldHeader-bgtext,
.h-fieldOrder-bgtext
	Field header and Number to the left of the field header		.h-fieldHeader-bgtext,
			.h-fieldOrder-bgtext {
			font-size: 10pt;
			background-color: #bbccdd;
			}

.h-page-text a:visited	Anchors inside the pageContent Layout component which have been visited	IE only.	.h-page-text a:visited {
			color: #666699;
			}

Edit Layout			

CSS selector	**Description**	**Notes**	**Standard styles in QuickPlace**

.h-fieldHeaderEdit-bgtext	Field header		.h-fieldHeaderEdit-bgtext {
			font-family: geneva, arial, sans-serif;
			font-size: 10pt;
			color: #000d40;
			background-color: #f4f9ff;
			}

.h-fieldEdit-text, .h-fieldEdit-text td	Field description text	Use exact selector, as shown	.h-fieldEdit-text, .h-fieldEdit-text td {
			font-family: geneva, arial, sans-serif;
			font-size: 10pt;
			color: #000d40;
			}

.h-fieldOrderEdit-bgtext, div .h-fieldOrderEdit-bgtext td	Number to the left of the field header	Use exact selector, as shown. All properties in this selector must be marked **! important** to take effect. E.g., **color: green ! important;**	.h-fieldOrderEdit-bgtext, div .h-fieldOrderEdit-bgtext td { /* Baroque selector is required for Netscape 4.x */
			font-family: geneva, arial, sans-serif ! important;
			font-size: 10pt ! important;
			font-weight: bold ! important;
			background-color: #000d4d ! important;
			padding-left: 5px ! important;
			padding-right: 3px ! important;
			color: #fc0 ! important;
			}

.h-fieldSmallEdit-text	Small field text	All properties in this selector must be marked **! important** to take effect.	.h-fieldSmallEdit-text {
			font-family: geneva, "ms sans serif", sans-serif ! important;
			font-size: 9pt ! important;
			}

CSS selector	Description	Notes	Standard styles in QuickPlace

.h-fieldSpecialEdit-text	Special field text	Used in Task Info field. All properties in this selector must be marked **! important** to take effect.	.h-fieldSpecialEdit-text {
			font-family: geneva, "ms sans serif", sans-serif ! important;
			font-size: 9pt ! important;
			color: #000d40 ! important;
			padding: 0px ! important;
			padding-right: 20px ! important;
			vertical-align: top;
			}

QuickSearch			

.h-searchField-text	Style of the text field associated with the quickSearch Layout component		.h-searchField-text {
			font-family: geneva, "ms sans serif", arial, sans-serif;
			font-size: 9pt;
			}

Classes defined by the default Theme The classes listed below are not built in to QuickPlace, but are defined by the default Theme's stylesheet. (Custom Themes are not required to use these classes, and are free to define any other classes as appropriate.) However if you are modifying the default Theme, you can modify these classes to get a particular effect. Note that in some cases you will need to define the class and its anchor context (a.classname) to get the desired result.			

.h-logo-text	Logo text		.h-logo-text, a.h-logo-text {
			text-decoration: none;
			}

.h-heading-textbg	Heading about table of contents and tools boxes		

.h-sidebar-bg	Background of table of contents and tool boxes		.h-sidebar-bg {
			font-family: sans-serif;
			font-size: 8pt; /* Height of whitespace in sidebar */
			background-color: #f7efbd;
			}

CSS selector	Description	Notes	Standard styles in QuickPlace

.h-toc-text, .h-tocSelected-text
 	
.h-toc-text
.h-tocSelected-text
	Text of item listed in table of contents and Text of selected item listed in table of contents	In the automatic generated stylesheet these styles are generated twice. See the Standard styles in QuickPlace.	.h-toc-text, .h-tocSelected-text {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 9pt;
			}
			.h-toc-text {
			color: #004573;
			}
			.h-tocSelected-text {
			color: #e74d4a;
			}

.h-nav-text, .h-tool-text, .h-signIn-text	Navigation link , Tool link and Sign In link		.h-nav-text, .h-tool-text, .h-signIn-text {
			font-family: geneva, "ms sans serif", arial, helvetica, sans-serif;
			font-size: 9pt;
			color: #004573;
			text-transform: lowercase;
			}

.h-actionButtonBorder-bg	Border of action button		.h-actionButtonBorder-bg {
			background-color: #6095b3;
			}

.h-actionButton-bg	Background of action button		.h-actionButton-bg {
			background-color: #f7eb7b;
			}

.h-actionButton-text	Text of action button		.h-actionButton-text {
			font-family: geneva, arial, sans-serif;
			font-size: 9pt;
			font-weight: bold;
			color: #004573;
			}

.h-actionSpace-text	Space between action buttons		.h-actionSpace-text {
			font-size: 9pt;
			}

.h-pageTitle-textbg	Background of Page title		

.h-pageAuthorMod-text	AuthorAndModified text		.h-pageAuthorMod-text {
			font-size: 9pt;
			}

CSS selector	Description	Notes	Standard styles in QuickPlace

.h-revision-text,
.h-revisionSelected-text
	Revision link (draft | published) and Selected revision link		.h-revision-text,
			.h-revisionSelected-text {
			font-size: 9pt;
			color: #004573;
			text-transform: capitalize;
			}

.h-accent-bg	Accent color	E.g., used in rule at bottom of page	.h-accent-bg {
			background-color: #e74d4a;
			}

.h-headlineFolderTab-bg	Background of unselected tab in headline folder		.h-headlineFolderTab-bg {
			background-color: #f7efbd;
			}

.h-headlineFolderTab-text	Text of unselected tab in headline folder		.h-headlineFolderTab-text {
			font-size: 9pt;
			}

.h-headlineFolderTabSelected-b
g
	Background of selected tab in headline folder		.h-headlineFolderTabSelected-b
			g {
			background-color: #004573;
			}

.h-headlineFolderTabSelected-te
xt
	Text of selected tab in headline folder		.h-headlineFolderTabSelected-te
			xt {
			font-size: 9pt;
			color: White;
			font-weight: bold;
			}

.h-edit-bg	Background of edit layout “docket”		.h-edit-bg {
			background-color: #d1def0;
			font-size: 8pt; /* Height of text in tab */
			}

.h-actionButtonEdit-text	Text of action button in edit layout		.h-actionButtonEdit-text {
			font-family: geneva, arial, helvetica, sans-serif;
			font-size: 10pt;
			color: #004573;
			font-weight: bold;
			text-transform: uppercase;
			}

CSS selector	Description	Notes	Standard styles in QuickPlace

.h-actionButtonBorderEdit-bg	Border of action button in edit layout		.h-actionButtonBorderEdit-bg {
			background-color: #000;
			}

.h-actionButtonEdit-bg	Background of action button in edit layout		.h-actionButtonEdit-bg {
			background-color: #fc0;
			}

.h-shadow-bg	Shadow	Used in sidebar and in edit layout “docket” shape	.h-shadow-bg {
			background-color: #666666;
			background-image: none;
			}

.h-shadowCorner-bg	“Missing” corner of shadow area		.h-shadowCorner-bg {
			background-color: white;
			}

4.2.3 Style Sheet previewer

Together with the downloadable files for this redbook, we have included a Style Sheet preview file. It contains hardcoded HTML representing many of the QuickPlace components, and includes the default QuickPlace Style Sheet, as well as the StyleSheet being developed.

By using the *stylesheetpreview.htm* file, you can quickly see how some of the parts in your QuickPlace Theme will look as you change the Style Sheet selectors. See Appendix L, “Additional Web material” on page 425 for instructions on how to get the previewer.

4.3 The QuickPlace Theme layout architecture

In this section wedescribe all the different components you can use in your layout files. As discussed in the beginning of the chapter, you can specify individual layout files for the following:

- Page in read mode
- Page in edit mode
- List folder
- Headlines folder
- Slideshow folder

For example, the Millennia layout in Figure 22 on page 49 is specified in the layout file for Page in read mode.

In most cases, you can use a single layout file to customize the look of page, list folder, and slideshow folder.

Table 5 table shows the components you can customize for each layout:

Table 5. Components you can customize

Component name	Page	Page Edit	List folder	Slide-show folder	Head-lines folder
Actions	x	x	x	x	x
AdvancedSearch	x		x	x	x
AuthorAndModified	x		Note 3	x	x
Chat	x		x	x	x
HeadlinesFolder					x
Help	x	x	x	x	x
Jump	Note 3		x	x	Note 2
Logo	x	x	x	x	x
Navigation	x		x	x	Note 2
Notify	x		x	x	x
Offline	x		x	x	x
PageContent	x	x	x	x	x
PageTitle	x	x	x		Note 1
Path	x		x	x	x
Print	x		x	x	x
QuickSearch	x		x	x	x
Revision	x		Note 3	x	x
SignIn	x		x	x	x
TOC	x		x	x	x
Tutorial	x		x	x	x
WhatsNew	x		x	x	x

Notes:

1. Although you can optionally include the PageTitle component in a Headlines folder, you would normally omit this component and display the page title prominently instead.
2. Do not use the Navigation and Jump components in the Headlines Folder layout because the Headlines Folder is designed to provide a headlines style of navigation in place of the previous/next navigation used in other folder types.
3. You can include the Jump component in the Page layout and you can include the AuthorAndModified and Revision components in the ListFolder layout. These components will all display as empty, using the HTML parameter emptyFormat.

The HTML tag that controls the style and placement of elements in a QuickPlace layout is the *<QuickPlaceSkinComponent tag>*. The basic syntax for the *<QuickPlaceSkinComponent>* tag is as follows:

Syntax

```
<QuickPlaceSkinComponent
name=<skincomponentname> (required)
format={<format html>} (optional)
selectedformat={<format html>} (optional)
emptyformat={<html>} (optional)
delimiter={<html>} (optional)
prefixHTML={<html>} (optional)
postfixHTML={<html>} (optional)
replaceString={String_1=Replacement_1 && String_2=Replacement_2 && ...}
(optional)
>

<table width="165" border="0" cellpadding=0 cellspacing=0>
   <tr>
      <td><img src="ecblank.gif" border=0 height=1 width=1></td>
         <QuickPlaceSkinComponent
         name=TOC
         format={<img src=a_right.gif border=0 height=7 width=7> <Item
class=h-toc-text>}
         selectedFormat={<img name=arightsel src=a_red_right.gif border=0
height=7 width=7> <Item class=h-tocSelected-text>}
         delimiter={<p>}
         emptyFormat={}
         replaceString={}
         >
         <img src="ecblank.gif" border=0 height=5 width=1 align=right>
      </td>
   </tr>
</table>
```

Note: Although you can use "<html>" (quotation marks) instead of {<html>} (curly brackets), we do not do this in our examples because quotation marks can be used in the HTML within the tag, and this cannot be done without using curly brackets.

```
format={<Item class="h-pagetitle-text">}
```

4.3.0.1 Layout component or Skin component

The QuickPlace tag `QuickPlaceSkinComponent` refers to Skin components. The word *Skin* is often confused with a full QuickPlace Theme. We will refer to *Layout component* in this redbook because this name better states what we use it for.

4.3.1 Explanation

name
Specifies the name of the Layout Component.

format
The keyword Item is replaced for each relevant entry.

selectedFormat
Same as format,but it applies to the selected Item value.

emptyFormat
What is returned when there are no values to iterate over.

delimiter
The HTML placed between each of the items in a list of values.

prefixHTML
The HTML placed before each of the values in a list.

postfixHTML
The HTML placed after each of the values in a list.

replaceString
Finds and replaces one or more strings with replacement strings.

4.3.2 Layout components

In this section we describe all the QuickPlace layout components and show (where relevant) how they look using our Millenia Theme Style Sheet.

4.3.2.1 Actions

The Actions component is used in all layouts.

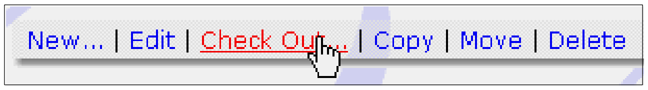


Figure 29. Actions

Actions attributes that are applicable

The following attributes are relevant to the Actions Layout component:

- name
- format
- emptyFormat
- delimiter
- prefixHTML

  Not often used with this Layout Component.
- postfixHTML

  Not often used with this Layout Component.
- replaceString

Strings available to replace:

```
add/remove members
back
begin install
cancel *
cleanup
change basics
change password
check in...
check in now
check out...
check out now
copy
customize
delete
delete folder **
delete theme
done * **
edit
folder...
groups
home
install now!
move
move room
new...
new calendar page
new room
new group
new page
new placebot
new room
new revision
new task page
new theme
next
previous
publish *
publish as *
reorder forms
room index
room options
room security
respond
save *
show/hide forms
```

Notes:

* Only show in Page Edit layout

**Only show in List Folder layout

Actions attributes that are not applicable

The following attributes are not relevant to the Actions Layout component.

- selectedFormat

  You never have an Actions Layout component selected.

Example:

```
<QuickPlaceSkinComponent
name=Actions
format={<Item class=action-text>}
emptyFormat={""}
```

```
delimiter={&nbsp;|&nbsp;}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
ReplaceString={Edit=<img src="edit.gif" width=20 height=10 alt="Edit"
border="0"> && Move=<img src="move.gif" width=20 height=10 alt="Move"
border="0">}
>
```

4.3.2.2 AdvancedSearch

AdvancedSearch is used in all the layouts except the Page Edit layout. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing.

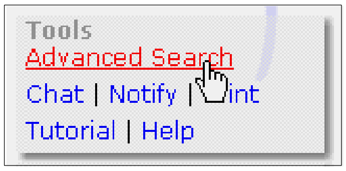


Figure 30. AdvancedSearch

AdvancedSearch attributes that are applicable

The following attributes are relevant to the AdvancedSearch Layout component:

- name
- format
- prefixHTML

  Not often used with this Layout Component.
- postfixHTML

  Not often used with this Layout Component.
- replaceString

Strings available to replace:

```
advanced search
```

AdvancedSearch attributes that are not applicable

The following attributes are not relevant to the AdvancedSearch Layout component:

- selectedFormat

  You never have an Actions Layout Component selected.

- emptyFormat

  Never empty.

- delimiter

  There is only one value in AdvancedSearch, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=AdvancedSearch
format={<Item class=h-tool-text>}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
ReplaceString={Advanced Search=<img src="asearch.gif" width=10 height=10
alt="Advanced Search" border="0">}
>
```

4.3.2.3 AuthorAndModified

AuthorAndModified shows in all the layouts except Page Edit and List folder. In List folder layout, it will display as empty, using the parameter emptyFormat.

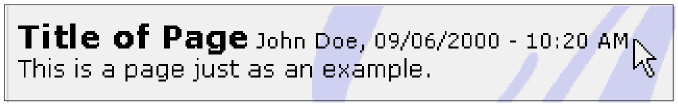

Title of Page
John Doe, 09/06/2000 - 10:20 AM
This is a page just as an example.

Figure 31. AuthorAndModified

AuthorAndModified attributes that are applicable

The following attributes are relevant to the AuthorAndModified Layout component:

- name
- format
- emptyFormat
- delimiter
- prefixHTML
- postfixHTML

AuthorAndModified attributes that are not applicable

The following attributes are not relevant to the AuthorAndModified Layout component:

- selectedFormat

You never have an Actions Layout Component selected

- replaceString

  You don't know what the values are. Author name and Modified date are always different.

Example:

```
<QuickPlaceSkinComponent
name=AuthorAndModified
format={<Item class=h-pageAuthorMod-text>}
emptyFormat={}
delimiter={, }
prefixHTML={<br><span class=authormod-text>Modified by: }
postfixHTML={</span>}
>
```

4.3.2.4 Chat

Chat can be used in all layouts except Page Edit. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing.

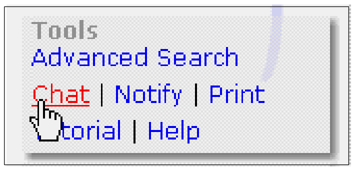


Figure 32. Chat

Chat attributes that are applicable

The following attributes are relevant to the Chat Layout component:

- name
- format
- prefixHTML

  Not often used with this Layout component.
- postfixHTML

  Not often used with this Layout component.
- replaceString

Strings available to replace:

```
chat
```

Chat attributes that are not applicable
The following attributes are not relevant to the Chat Layout componen:

- selectedFormat

  You never have an Chat Layout component selected.

- emptyFormat

  Never empty.

- delimiter

  There is only one value in Chat, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=Chat
format={<Item class=h-tool-text>}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
replaceString={Chat=<img src="chat.gif" width=10 height=10 alt="Chat"
border="0">}
>
```

4.3.2.5 HeadlinesFolder

HeadlinesFolder is only used in the Headlines folder layout.

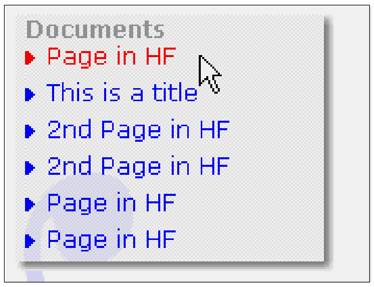


Figure 33. HeadlinesFolder

HeadlinesFolder attributes that are applicable
The following attributes are relevant to the HeadlinesFolder Layout component:

- name
- format
- selectedFormat
- delimiter

- prefixHTML

  Not often used with this Layout component.
- postfixHTML

  Not often used with this Layout component.
- replaceString

  Not often used with this Layout component since you can change the name of your pages inside QuickPlace.

Strings available to replace:

```
<Titles of your different pages>
```

HeadlinesFolder attribute that is not applicable

The following attribute is not relevant to the HeadlinesFolder Layout component:

- emptyFormat

  Never empty.

Example:

```
<QuickPlaceSkinComponent
name=HeadlinesFolder
format={<td><img src="headlineTabLeft.gif" width=20 height=20 alt=""
border="0"></td><td class=headline-bg nowrap><Item
class=headline-text></td><td><img src="headlineTabRight.gif" width=20
height=20 alt="" border="0"></td>}
selectedFormat={<td><img src="headlineSelectedTabLeft.gif" width=20
height=20 alt="" border="0"></td><td class=headlineSelected-bg nowrap><Item
class=headlineSelected-text></td><td><img
src="headlineSelectedTabRight.gif" width=20 height=20 alt=""
border="0"></td>}
delimiter={<td><img src="betweenHeadline.gif" width=5 height=20 alt=""
border="0"></td>}
prefixHTML={<td><img src="preHeadline.gif" width=10 height=20 alt=""
border="0"></td>}
postfixHTML={<td><img src="postHeadline.gif" width=10 height=20 alt=""
border="0"></td>}
replaceString={<PageTitle>=<img src="pagetitle.gif" width=100 height=50
alt="Page Title" border="0">}
>
```

4.3.2.6 Help

Help should be used in all layouts.

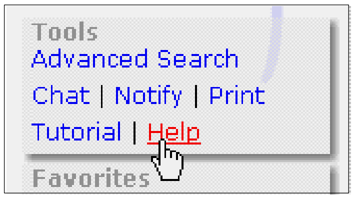


Figure 34. Help

Help attributes that are applicable

The following attributes are relevant to the Help Layout component:

- name
- format
- prefixHTML

  Not often used with this Layout Component.
- postfixHTML

  Not often used with this Layout Component.
- replaceString

Strings available to replace:

```
help
```

Help attributes that are not applicable

The following attributes are not relevant to the Help Layout componen:

- selectedFormat

  You never have an Help Layout component selected.
- emptyFormat

  Never empty.
- delimiter

  There is only one value in Help, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=Help
format={<Item class=h-toc-text>}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
replaceString={Help=<img src="help.gif" width=10 height=10 alt="Help"
border="0">}
```

4.3.2.7 Jump

Jump can be used in all Folder layouts. It brings up a JavaScript prompt window, allowing you to write the first letters of the column the folder is sorted by. Jump then takes you to the first document that starts with what you wrote. This is similar to the function of start to type in a Notes view in the Notes Client.

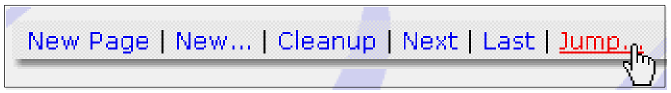


Figure 35. Jump

Jump attributes that are applicable

The following attributes are relevant to the Jump Layout component:

- name
- format
- emptyFormat
- prefixHTML
- postfixHTML
- replaceString

Strings available to replace:

`Jump...`

Jump attributes that are not applicable

The following attributes are not relevant to the Jump Layout component:

- selectedFormat

  You never have an Jump Layout component selected.

- delimiter

  There is only one value in Jump, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=Jump
format={<Item class=h-toc-text>}
emptyFormat={}
>
```

4.3.2.8 Logo

Logo shows in all layouts. Logo displays the name of the QuickPlace.

Figure 36. Logo

Logo attributes that are applicable

The following attributes are relevant to the Logo Layout component:

- name
- format
- prefixHTML
- postfixHTML
- replaceString
- Not often used with this Layout component, since you can change the logo that displays the name of your QuickPlace.

Strings available to replace:

```
<Name of QuickPlace>
```

Logo attributes that are not applicable

The following attributes are not relevant to the Logo Layout component:

- selectedFormat

  You never have an Logo Layout component selected.
- emptyFormat

  Never empty.
- delimiter

  There is only one value in Logo, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=Logo
format={<Item class=h-toc-text>}
>
```

4.3.2.9 Navigation

Navigation is used in all layouts except Page Edit and Headlines folder. The Headlines folder is designed to provide a headlines stye of navigation in place of the previous/next navigation used in other folder types.

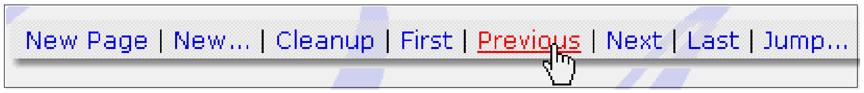

New Page | New... | Cleanup | First | Previous | Next | Last | Jump...

Figure 37. Navigation

Navigation attributes that are applicable

The following attributes are relevant to the Navigation Layout component:

- name
- format
- emptyFormat
- delimiter
- prefixHTML

  Not often used with this Layout component.
- postfixHTML

  Not often used with this Layout component.
- replaceString

Strings available to replace:

```
first
last
previous
next
folder
```

Navigation attribute that is not applicable

The following attribute is not relevant to the Navigation Layout component:

- selectedFormat

  You never have an Navigation Layout component selected.

Example:

```
<QuickPlaceSkinComponent
name=Navigation
format={<td><Item class=nav-text></td>}
emptyFormat={<td>&nbsp;</td>}>
delimiter={}
prefixHTML={<td><img src="pre.gif" width=13 height=13 alt=""
border="0"></td>}
postfixHTML={<td><img src="post.gif" width=13 height=13 alt=""
border="0"></td>}
```

```
ReplaceString={
First=<img src="first.gif" width=100 height=50 alt="First" border="0"> &&
Last=<img src="last.gif" width=100 height=50 alt="Last" border="0"> &&
Previous=<img src="previous.gif" width=100 height=50 alt="Previous "
border="0"> && Next=<img src="next.gif" width=100 height=50 alt="Next"
border="0"> && Folder=<img src="folder.gif" width=100 height=50
alt="Folder" border="0">}
>
```

4.3.2.10 Notify

Notify is used in all layouts except Page Edit. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing. Notify is a link to a form allowing you to send a notify to users about the page you were on.

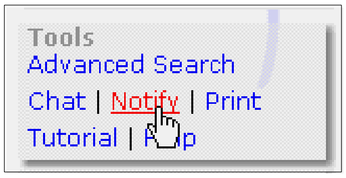


Figure 38. Notify

Notify attributes that are applicable

The following attributes are relevant to the Notify Layout component:

- name
- format
- prefixHTML

  Not often used with this Layout component.
- postfixHTML

  Not often used with this Layout component.
- replaceString

Strings available to replace:

`notify`

Notify attributes that are not applicable

The following attributes are not relevant to the Notify Layout component:

- selectedFormat

  You never have an Notify Layout component selected.

- emptyFormat

  Never empty.

- delimiter

  There is only one value in Notify, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=Notify
format={<Item class=h-toc-text>}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
replaceString={Notify=<img src="notify.gif" width=10 height=10 alt="Notify"
border="0">}
>
```

4.3.2.11 Offline

Offline is used in all layouts except Page Edit. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing. It gives the user the ability to go offline with the QuickPlace.

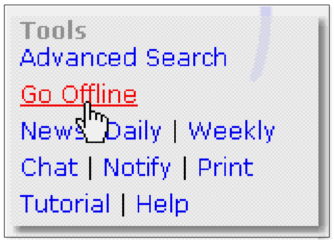


Figure 39. Offline

Offline attributes that are applicable

The following attributes are relevant to the Offline Layout component:

- name
- format
- prefixHTML

  Not often used with this Layout component.

- postfixHTML

  Not often used with this Layout component.

- replaceString

Strings available to replace:

`offline`

Offline attributes that are not applicable

The following attributes are not relevant to the Offline Layout component:

- selectedFormat

  You never have an Offline Layout component selected.

- emptyFormat

  Never empty.

- delimiter

  There is only one value in Offline, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=Offline
format={<Item class=h-tool-text>}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
replaceString={Offline=<img src="offline.gif" width=10 height=10
alt="Offline" border="0">}
>
```

4.3.2.12 PageContent

PageContent is used in all layouts. PageContent is where all the text that is written to the document is placed.

PageContent attribute that is applicable

The following attribute is relevant to the PageContent Layout component:

- name

PageContent attributes that are not applicable

The following attributes are not relevant to the PageContent Layout component:

- format

  Not applicable in PageContent.

- selectedFormat

  Not applicable in PageContent.

- emptyFormat

  Not applicable in PageContent.

- delimiter

  Not applicable in PageContent.
- prefixHTML

  Not applicable in PageContent.
- postfixHTML

  Not applicable in PageContent.
- replaceString

  Not applicable in PageContent.

Example:

```
<QuickPlaceSkinComponent
name=PageContent
>
```

4.3.2.13 PageTitle

PageTitle should be used in all layouts except SlideShow folder and Headlines folder. Although you can include it in a Headlines folder layout, you would normally omit this component and display the page title prominently instead.

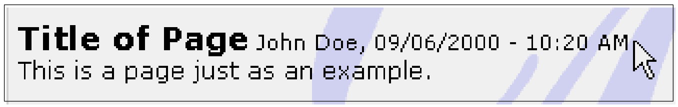

Title of Page John Doe, 09/06/2000 - 10:20 AM
This is a page just as an example.

Figure 40. PageTitle

PageTitle attributes that are applicable

The following attributes are relevant to the PageTitle Layout component:

- name
- format
- emptyFormat
- prefixHTML

  Not often used with this Layout component.
- postfixHTML

  Not often used with this Layout component.
- replaceString

Not often used with this Layout component, since you can change the name of your pages inside QuickPlace.

Strings available to replace:

```
<Titles of your different pages>
```

PageTitle attributes that are not applicable

The following attributes are not relevant to the PageTitle Layout component:

- selectedFormat

  You never have an PageTitle Layout component selected.

- delimiter

  There is only one value in PageTitle, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=PageTitle
format={<Item class=h-pageTitle-text>}
emptyFormat={}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
replaceString={<PageTitle>=<img src="pagetitle.gif" width=100 height=50
alt="Page Title" border="0">}
>
```

4.3.2.14 Path

Path is used in all layouts except Page Edit. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing. Path shows what folder in the QuickPlace you are in at the moment.

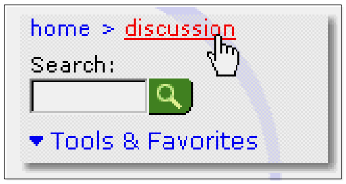


Figure 41. Example of Path

Path attributes that are applicable

The following attributes are relevant to the Path Layout componen:

- name
- format

- emptyFormat
- delimiter
- prefixHTML

  Not often used with this Layout component.
- postfixHTML

  Not often used with this Layoutcomponent.
- replaceString

  Not often used with this Layout component, since you can change the name of your folders inside QuickPlace

Strings available to replace:

```
home and <Titles of your different folders>
```

Path attribute that is not applicable

The following attribute is not relevant to the Path Layout component:

- selectedFormat

  You never have an Path Layout component selected.

Example:

```
<QuickPlaceSkinComponent
name=Path
format={<Item class=h-nav-text>}
emptyFormat={}
delimiter={&gt;}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
replaceString={<FolderTitle>=<img src="foldertitle.gif" width=100
height=50 alt="Folder Title" border="0">}
>
```

4.3.2.15 Print

Print is used in all layouts except Page Edit. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing. This link brings up a window with only the PageTitle and PageContent.

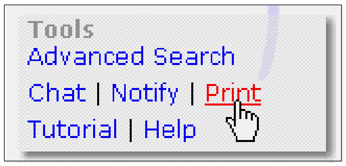


Figure 42. Print

Print attributes that are applicable

The following attributes are relevant to the Print Layout component:

- name
- format
- prefixHTML

  Not often used with this Layout component.
- postfixHTML

  Not often used with this Layout component.
- replaceString

Strings available to replace:

```
print
```

Print attributes that are not applicable

The following attributes are not relevant to the Print Layout component:

- selectedFormat

  You never have an Print Layout component selected.
- emptyFormat

  Never empty
- delimiter

  There is only one value in Print, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=Print
format={<Item class=h-tool-text>}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
replaceString={print=<img src="print.gif" width=50 height=10 alt="Print"
border="0">}
>
```

4.3.2.16 QuickSearch

QuickSearch is used in all layouts except Page Edit. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing. Although you cannot specify any format for QuickSearch, you can change the class .h-searchField-text in the StyleSheet to change the look of QuickSearch.

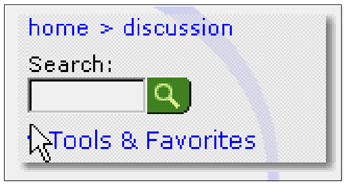


Figure 43. QuickSearch

QuickSearch attribute that is applicable

The following attribute is relevant to the QuickSearch Layout component:

- name

QuickSearch attributes that are not applicable

The following attributes are not relevant to the QuickSearch Layout component:

- format

  Not applicable in QuickSearch.
- selectedFormat

  Not applicable in QuickSearch.
- emptyFormat

  Not applicable in QuickSearch.
- delimiter

  Not applicable in QuickSearch.
- prefixHTML

  Not applicable in QuickSearch.
- postfixHTML

  Not applicable in QuickSearch.
- replaceString

  Not applicable in QuickSearch.

Example:

```
<QuickPlaceSkinComponent
name=QuickSearch
>
```

4.3.2.17 Revision

Revision is used in all layouts except Page Edit and List folder. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing. It shows whether a document is in revision. It does this by showing the Draft in Progress and Published Version links.

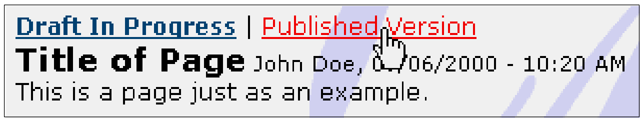

Draft In Progress | Published Version
Title of Page John Doe, 0./06/2000 - 10:20 AM
This is a page just as an example.

Figure 44. Revision

Revision attributes that are applicable

The following attributes are relevant to the Revision Layout component:

- name
- format
- selectedFormat
- emptyFormat
- delimiter
- prefixHTML
- postfixHTML
- replaceString

Strings available to replace:

```
   draft in progress
   published version
```

Example:

```
<QuickPlaceSkinComponent
name=Revision
format={<Item class=h-revision-text>}
selectedFormat={<Item class=h-revisionSelected-text>}
emptyFormat={}
delimiter={&nbsp;|&nbsp;}
prefixHTML={<tr><td width=100% valign=top class=h-revision-text>}
postfixHTML={</td></tr>}
```

```
replaceString={
   draft in progress=<img src="dip.gif" width=100 height=50 alt="Draft in
Progress" border="0"> &&
   published version=<img src="pv.gif" width=100 height=50 alt="Published
Version" border="0">}
>
```

4.3.2.18 SignIn

SignIn is used in all layouts except Page Edit. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing. This component only shows if you allow [Anonymous] users in to the QuickPlace.

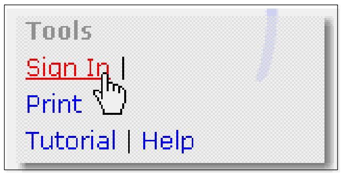


Figure 45. SignIn

SignIn attributes that are applicable

The following attributes are relevant to the SignIn Layout component:

- name
- format
- prefixHTML

  Not often used with this Layout component.
- postfixHTML

  Not often used with this Layout component.
- replaceString

Strings available to replace:

```
   signin
```

SignIn attributes that are not applicable

The following attributes are not relevant to the SignIn Layout component:

- selectedFormat

  You never have a SignIn Layout component selected.
- emptyFormat

  Never empty.

- delimiter

  There is only one value in SignIn, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=SignIn
format={<Item class=h-signin-text>}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
replaceString={signin=<img src="signin.gif" width=50 height=10 alt="Sign
In" border="0">}
>
```

4.3.2.19 TOC

TOC is used in all ayouts. It shows allfolders, rooms and pages you selected to show there.

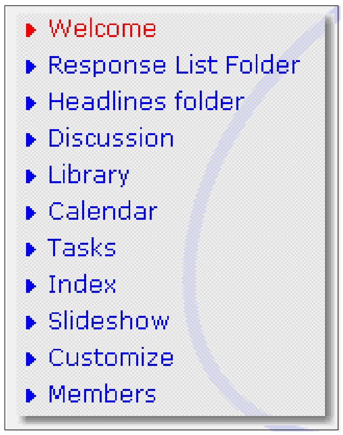


Figure 46. TOC

TOC attributes that are applicable

The following attributes are relevant to the TOC Layout component:

- name
- format
- selectedFormat
- emptyFormat
- delimiter
- prefixHTML

- postfixHTML
- replaceString

  Not often used with this Layout component, since you can change the name of your pages and folders inside QuickPlace

Strings available to replace:

```
<Titles of your different folders and pages>
```

Example:

```
<QuickPlaceSkinComponent
name=TOC
format={<td class=h-toc-text><Item class=h-toc-text></td>}
emptyFormat={}
delimiter={}
prefixHTML={<tr>}
postfixHTML={</tr>}
replaceString={<FolderTitle/PageTitle>=<img src="foldertitle.gif"
width=100 height=50 alt="Folder Title" border="0">}
>
```

4.3.2.20 Tutorial

Tutorial is used in all layouts except Page Edit. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing.

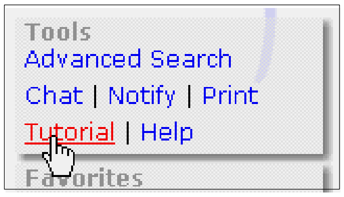


Figure 47. Tutorial

Tutorial attributes that are applicable

The following attributes are relevant to the Tutorial Layoutcomponent:

- name
- format
- prefixHTML

  Not often used with this Layout component.
- postfixHTML

  Not often used with this Layout component.

- replaceString

Strings available to replace:

`tutorial`

Tutorial attributes that are not applicable

The following attributes are not relevant to the Tutorial Layout componen:

- selectedFormat

  You never have an Tutorial Layout component selected.
- emptyFormat

  Never empty
- delimiter

  There is only one value in Tutorial, so no delimiter is applicable.

Example:

```
<QuickPlaceSkinComponent
name=Tutorial
format={<Item class=h-tool-text>}
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
replaceString={tutorial=<img src="tutorial.gif" width=50 height=10
alt="Tutorial" border="0">}
>
```

4.3.2.21 WhatsNew

WhatsNew is used in all layouts except Page Edit. It can be used in the Page Edit layout, but is usually omitted from it to give more room for editing. WhatsNew are links to what is new in this QuickPlace, either today or the last week.

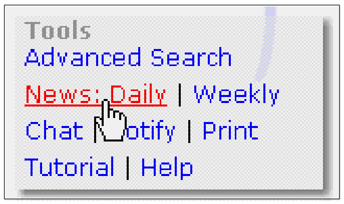


Figure 48. WhatsNew

WhatsNew attributes that are applicable

The following attributes are relevant to the WhatsNew Layout component:

- name
- format
- emptyFormat
- delimiter
- prefixHTML

  Not often used with this Layout component.
- postfixHTML

  Not often used with this Layout component.
- replaceString

Strings available to replace:

```
   news: daily
   weekly
```

WhatsNew attribute that is not applicable

The following attribute is not relevant to the WhatsNew Layout component:

- selectedFormat

  You never have an WhatsNew Layout component selected.

Example:

```
<QuickPlaceSkinComponent
name=WhatsNew
format={<Item class=h-tool-text>}
emptyFormat={}
delimiter={ / }
prefixHTML={<img src="pre.gif" width=13 height=13 alt="" border="0">}
postfixHTML={<img src="post.gif" width=13 height=13 alt="" border="0">}
replaceString={
   news: daily=<img src="dailynews.gif" width=50 height=10 alt="News Daily"
border="0"> &&
   weekly=<img src="weeklynews.gif" width=50 height=10 alt="News Weekly"
border="0">}
>
```

4.3.3 Images in Themes

To use images in Themes, make sure that the images are located so they are correctly referenced in the HTML layout files. An easy way to accomplish this is to put your images in the same directory (or folder) as the layouts and Style

Sheet.You reference the images without paths and QuickPlace will upload these images automatically. Putting your images, layouts and Style Sheet together in the same directory (or folder) is also a good way of keeping your different Themes organized.

Example in HTML:

```
<img src="ecblank.gif" border=0 height=5 width=1 align=right>
```

Example in Style Sheet

```
.h-shadTopRight-bg {
   background-image: url(ShadTopRight.gif);
   background-repeat: no-repeat;
}
```

Note: While it is convenient to have everything in the same directory, it is not required as long as there is a valid reference to your images. For example, if you have your images in a directory named *images* that is on the same level as your layout file directory, you would reference your image like this:

```
<img src="..\images\ecblank.gif" border=0 height=5 width=1 align=right>
```

For additional discussion of how to work with images and JavaScript in Themes plus other considerations, see 8.2.19, "QuickPlace Object Model and HTML: Building URLs" on page 246.

This concludes our description of the QuickPlace layout components. We will now describe how we added extra functionality to the Millennia Theme through the Favorites page.

4.4 Adding the Favorites HTML page

The Favorites page gives users the ability to maintain a list of favorite pages in the Place layout. It extends the Millennia Theme by providing a way of editing your page Favorites Cookie list.

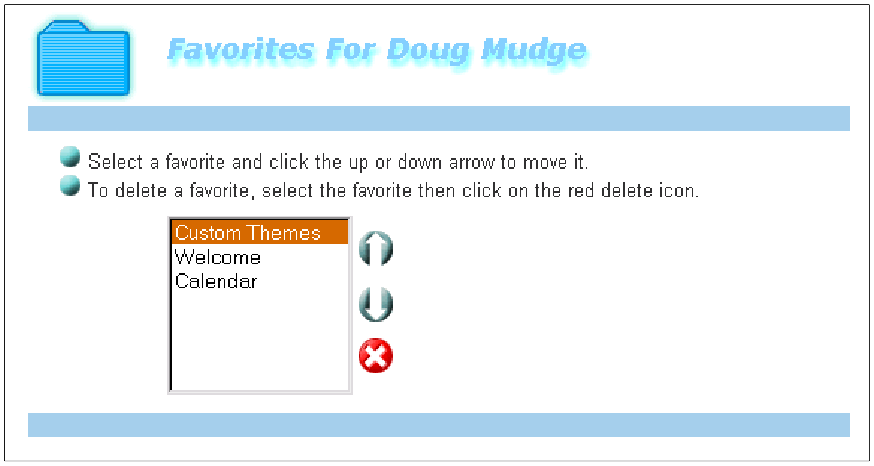
The Favorites HTML page files are available for download from the IBM Redbooks Web site. See Appendix L, "Additional Web material" on page 425 for instructions on how to get the sample files.

The purpose of this example is not to teach how to write HTML or JavaScript. It is designed purely to show how to implement HTML and JavaScript into QuickPlace. For this reason, we will only look at the code exposing QuickPlace features through HTML and JavaScript.

The Favorites HTML page uses cookies to save the links. Cookies are a convenient way of saving variables in persistent state. This means that your browser caches the values onto a small file on your hard disk, not in RAM. The advantage of this is that you can close your browser, switch off your computer, and the information will still be there when you start your browser again.

Web sites can only see cookies that were created by them. For example, if you have a cookie from www.amazon.com, it cannot be read by www.cnn.com. The only downside to this is that your cookies will only work on the server where you created them.

Note that the *Favorites For Doug Mudge* image is generated on the fly, when the user's name is Doug Mudge. The Page looks like this when installed:



Favorites For Doug Mudge

Select a favorite and click the up or down arrow to move it.
To delete a favorite, select the favorite then click on the red delete icon.

Custom Themes
Welcome
Calendar

Figure 49. The Favorites HTML Page

Installing Favorites

1. Open a QuickPlace to which you have Editor access in your Web browser.
2. Click the **New...** button.
3. Choose **Imported Page**, click **Next**, and the *Edit Page* appears.
4. Enter the name of the Page as **Edit Favorites**. You *must* name it this if you are using the Favorites tool in the Millennia Theme.
5. Deselect the **Show the title, author and date on page?** check box.
6. Click the **Browse** button, and the Open dialog box appears.
7. Open the **EditFavorites.htm** file and click **Publish**.
8. Choose where you want to put this Page. We suggest the **Index**, if you are using the Millennia Theme, because it will already show up in the Favorites Toolbox.

9. Choose **Next...** and the file and its linked images are uploaded.
10. A glowing text will appear in the middle of the page, including the user's name. If you cannot see this, make sure that you have followed all of the previous steps.

The Favorites tool works with the Favorites component. The Favorites component is available as a separate file that you can cut and paste into any Web application, as well as a part of the Millennia Theme.

Start the page off by creating a series of images on the page to force QuickPlace to upload them. This is done by rendering them as 1x1pixel images. We will need them later in the JavaScript, so we need to render them here and access them later in the document:

```
<img name=ReorderDown2 src=ReorderDown.gif width="1" height="1" border=0>
<img name=ReorderDownHL src="ReorderDownHiLite.gif" width="1" height="1" border=0>
<img name=ReorderUp2 src=ReorderUp.gif width="1" height="1" border=0>
<img name=ReorderUpHL src="ReorderUpHiLite.gif" width="1" height="1" border=0>
<img name=Delete2 src=Delete.gif width="1" height="1" border=0>
<img name=DeleteHL src="DeleteHiLite.gif" width="1" height="1" border=0>
```

In the document there is an image of a folder with text next to it, displaying the user's name in a GIF image. To render this, we have used three technologies implemented with QuickPlace:

- CGI Variables, to retrieve the user's name.
- JavaScript, to perform error trapping edit the name and present the page.
- The QuickPlace Graphics Server, to render the image.

The first thing we do in the fnPrintName function is check that the REMOTE_USER variable has been declared. The REMOTE_USER CGI variable is declared and a value assigned to it automatically in QuickPlace pages.

If it has not been declared, it means the file is not being accessed while being run from a QuickPlace. In this case, the Graphics server (read more about the Graphics server in Chapter 5, "The Graphics server" on page 111) will not be able to engage and we will see a blank image.

To avoid this, we test for the value and if it is not found, then just write some plain text into the area where the image normally appears, as follows:

```
if( typeof REMOTE_USER == 'undefined'){
    var sImgSRC = '<b>Edit Favorites</b><br><br>'
} else {
```

The following lines of code prepare a URL which will be sent to the server via an SRC URL. The server creates a gif image and sends back the file. This form of technology uses the Domino Graphics Server.

Prepare the URL:

```
var sImgSRC = '<img src=../../Main.nsf?';
```

Now engage the Graphics Server by calling the GetImage URL:

```
sImgSRC += 'GetImage&TextString=Favorites%20For%20';
```

Bring in the user’s name, apply formatting in the fnStringFormat command, and escape any non-ASCII characters:

```
sImgSRC += escape( fnStringFormatt( REMOTE_USER ) );
```

Turn on the Effect of Shadow, set the font name, size and set it to be bold italic:

```
sImgSRC += '&EffectType=Shadow&FontBold=1&FontItalic=1';
sImgSRC += '&FontName=Verdana&FontPointSize=14';
```

Set the font color, the opacity and background parameters:

```
sImgSRC += '&fontColor=80CCFF&Opacity=100&BkColor=FFFFFF';
```

Set blurring and the shadow color, then set the X and Y offset parameters. These digits represent how many characters left and right the shadow is to be away from the main text:

```
sImgSRC += '&blurFactor=5&shadowColor=80FFFF';
sImgSRC += '&ShadowXOffset=2&ShadowYOffset=4 border=0>';
```

Write the text or image (if you are on a QuickPlace):

```
document.write( sImgSRC )
```

Further down we see a groovy script that you will almost certainly need if you are working with QuickPlace in JavaScript. The USER_NAME CGI variable is returned in the hierarchical format, for example:

```
CN=Doug Mudge/OU=QuickPlace/OU=QP/O=Server
```

To display this as a Common Name, we use the following function to strip away everything before the first equal (=) sign and after the first backslash (/).

```
function fnStringFormatt(sTxt) {
    //this function returns the common name of a person
    //from a full Domino canonical name.
    iTxtStart = sTxt.indexOf('=');
    iTxtStart++;
    iTxtEnd = sTxt.indexOf('/');
    if(iTxtEnd == -1) iTxtEnd = sTxt.length;
    sTxt = sTxt.substr(iTxtStart,iTxtEnd - iTxtStart);
    return sTxt;
```

That is the end of the functions for writing the title. Now that everything is prepared, it is time to write it to the document, thus making it visible:

```
<script language=JavaScript>fnPrintName();</script>
```

In the lower part of the page we see where the main select is created. This too is written out using JavaScript. It is written out dynamically to allow the cookies to be fed into the script:

```
<select name="selMain" colwidth=40 size="7">
   <script>fnWriteFavSelect()</script>
</select>
```

In the lower part of the Page we see where the main select is created:

```
<script>document.forms.formFavorites.selMain.selectedIndex = 0;</script>
```

Then come the links to move the elements in the selection list up and down; this is one of three. Also note that there is also a MouseOver being run on this anchor tag:

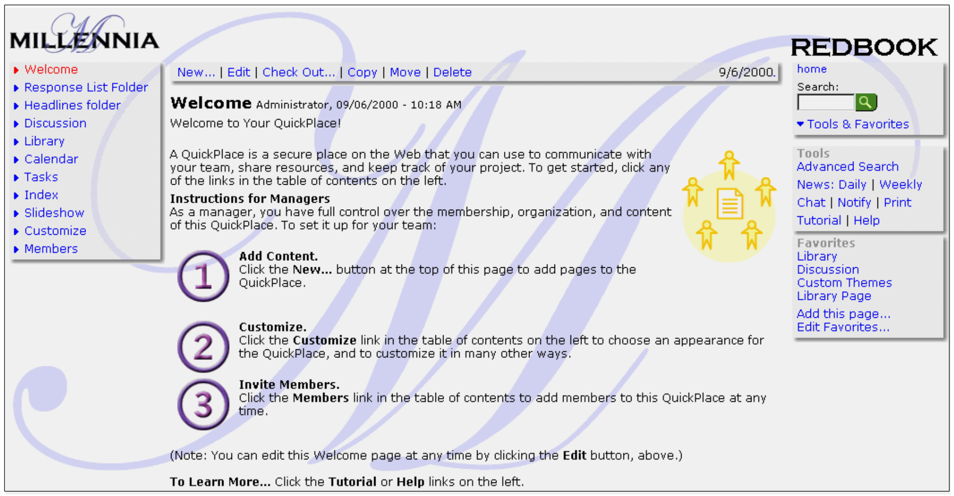
```
<a href="javascript:void(0)"
   onClick="javascript:fnMoveUpOrDown('up');return false;"
   onmouseOver="ReorderUp.src=ReorderUpHL.src"
   onmouseOut="ReorderUp.src=ReorderUp2.src"><img name=ReorderUp
   alt="Move favorite up" src=ReorderUp.gif width=23 height=24 border=0>
</a>
```

Lines after this close out the tables and end the page.

4.4.1 The finished Millennia Theme

Figure 50 on page 109 shows our main theme for the Millennia company. As you can see, it has a unique look that is different from the default QuickPlace

Theme. The advantage of the QuickPlace layout is that behind it, there is all the flexibility that QuickPlace has to offer.



Welcome Administrator, 09/06/2000 - 10:18 AM

Welcome to Your QuickPlace!

A QuickPlace is a secure place on the Web that you can use to communicate with your team, share resources, and keep track of your project. To get started, click any of the links in the table of contents on the left.

Instructions for Managers
As a manager, you have full control over the membership, organization, and content of this QuickPlace. To set it up for your team:

1 **Add Content.**
Click the **New...** button at the top of this page to add pages to the QuickPlace.

2 **Customize.**
Click the **Customize** link in the table of contents on the left to choose an appearance for the QuickPlace, and to customize it in many other ways.

3 **Invite Members.**
Click the **Members** link in the table of contents to add members to this QuickPlace at any time.

(Note: You can edit this Welcome page at any time by clicking the **Edit** button, above.)

To Learn More... Click the **Tutorial** or **Help** links on the left.

Figure 50. The Millennia QuickPlace Theme

We created this Theme to illustrate many of the opportunities QuickPlace offers. In our testing, we discovered things we would have done differently, and we have included things in the Theme to show it can be done (without necessarily recommending that you create your Theme exactly like the Millennia Theme). Following are a few things to consider when creating your own Theme:

- While the stylistic *M* background image in the Millennia Theme gives a very appealing experience when entering the Place, you should consider whether it inhibits reading the documents in the Place if your users will do a lot of reading.
- When you add functionality like the Favorites page, make sure that everybody understands that you now have a Theme that can have functional bugs, and that probably requires more maintenance than a more simple Theme.

  For example, the Favorites page utilizes cookies, which means that if you have users that work on different PCs, they will not be able to have one common Favorites list. Your users may be so fond of your Favorites list that they will rebel unless you can give them one common list of favorites, and that is probably a much bigger job than the implementation we have shown you.

- In the Millennia Theme, we placed the Actions component at the *top* of the page. In contrast, the default QuickPlace Theme has these at the *bottom* of the page (see the list of actions in 4.3.2.1, “Actions” on page 79 - one example is *Next*).Also, some of the QuickPlace system pages *New...* have a hardcoded reference to the Actions located at the botton of the page, such as the following:

  ```
  Click the Next button below when you have finished filling out this
  form.
  ```

  While this is not a significant issue, it can be confusing for new users.

- Where your Theme is used in Places that allow anonymous reader access, very few Actions are shown. Instead, the SignIn component is shown somewhere else in the layout. We found, with our Millennia Theme, that users that were listed as author members initially had difficulty finding the SignIn link and got frustrated because they couldn’t create documents (as they only has anonymous access rights).

  As a solution, we could add more emphasis to the SignIn component, or include hints on the Welcome page like the following:

  ```
  Sign in to edit and create new documents
  ```

There are many different ways to do different things, and the best way to find out what works is to ask real users to perform common tasks with a prototype of your Theme and learn from that.

4.5 Summary

In this chapter we discussed what a QuickPlace Theme is and how to create one. To create a custom Theme, you normally need at least one HTML layout file and a Style Sheet file.

We also described the Style Sheet selectors that QuickPlace uses, and all the layout components you can use in your custom Theme. Finally, we showed you how you can add additional functionality to a Theme by adding a Favorites list to the Theme.

Chapter 5. The Graphics server

The Graphics server is part of the QuickPlace server and is used to produce graphics through two basic functions with various variations. The first function is to produce a graphic image from a text string, and the other is to combine two existing images in some fashion to create a new image.

This is a very powerful feature because it allows you to create advanced graphic images on the fly that includes variable information like the user's name.

In this chapter we discuss the various results you can achieve, as well as what to avoid and why to avoid it.

Images are created using a URL syntax consisting of the *?GetImage* call together with different parameters.

Example:

```
<img scr="http://yourserver_name/yourdatabase_name?GetImage
&TextString=Millennia+Redbook">
```

For a simpler way of showing an image on a page where the image is already an attachment and you want to make the URL dynamic to all databases, use:

```
<img src="?GetImage&TextString=Millennia+Redbook">
```

For simplicity, we will use this as the example code for the rest of this chapter.

5.1 Parameter overview

When you create a GetImage URL, it basically consists of your source text and a number of parameters that specify how you want your output to look. Each parameter gives a certain functionality. Internally, the Graphics server has organized different types of functionality into different objects (like the input text object LRTextObj, the font object LRFontObj, and so on). The parameters you specify will be executed as methods against the object they belong to in the Graphic Server.

Although you do not address the object directly, we will organize our description of the parameters according to which internal object they belong to.

Note: All object names start with *LR* because the code name of the Graphic server was *LimeRick*.

In the following sections, we list all available object names and describe their individual parameters. If there are any considerations regarding an object or parameter, we provide those below each object table.

5.1.0.1 LRMethod

As shown in Table 6, this parameter specifies the function that the Graphics server is to provide.

Table 6. LRMethod parameters

Method	Description	Default value
Method	The method designer wishes to call; supported ones are: **FromText** - generate the image for a text string. **Merge** - generate image by combining two images.	FromText

Example:

```
<img src=?GetImage&Method=FromText&TextString=Millennia+Redbook>
```

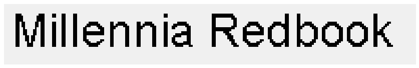


Figure 51. Example of Method FromText in LRMethod

```
<img src=?GetImage&Method=MergeImage
&BaseImage=/quickplace/redbook/Main.nsf/$All/Limerick/$FILE/m.gif?OpenElem
ent&OverlayImage=/quickplace/redbook/Main.nsf/$All/Limerick/$FILE/RedTrans
p.gif?OpenElement>
```

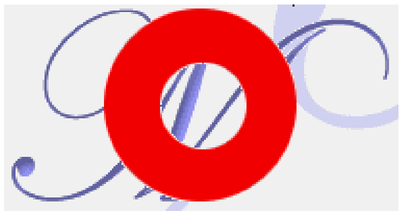


Figure 52. Example of Method Merge in LRMethod

5.1.1 Method: FromText

The method to create an image from a text string.

5.1.1.1 LRTextObj

Table 7 shows and describes the text string to use for creating an image.

Table 7. LRTextObj parameters

Method	Description	Default value
TextString	The text string of the object.	NULL

Example:

```
<img src=?GetImage&TextString=Millennia+Redbook>
```

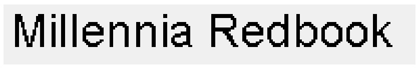


Figure 53. Example of LRTextObj

5.1.1.2 LROutputObj

Table 8 shows and describes the output operations to apply to the final image.

Table 8. LROutputObj parameters

Method	Description	Default value
OutRectWidth	This specifies the width in pixels of the output rectangle.	0
OutRectHeight	This specifies the height in pixels of the output rectangle.	0

Method	**Description**	**Default value**
OutOp	If a value other than NONE is specified, then at least one of OurRectWidth OR OutRectHeight must be specified. In other words, if both OurRectWidth and OutRectHeight ARE 0, then OutOp is NONE! Valid choices are: **NONE** - no operation **CLIP_ELLIPSIS** - clip image to the specified rect, if the text will not fit, then truncate text and append ellipsis "..." **CLIP_WORDWRAP** - clip image to the specified rect, if the text will not fit, word wrap to fit. **CLIP_WWEQUAL** - similar to CLIP_WORDWRAP but this will try to make the wrapped lines equal in length. **CLIP_TEXTBOUND** - this is used only when there is a BkImage input. This will cause the output image to be clipped to the text boundary.	Default is 0 *unless* a value is specified for either OutRectWidth orOutRectHeight, then the default is CLIP_ELLIPSIS
OutMaxLines	This specifies the maximum limit for the number of lines of word wrapped text that we will produce in conjunction with the OutOp=CLIP_WORDWRAP. If the OurRectHeight is specified, then OutRectHeight will take precedence if the the number of lines specified will not fit.	Default is 0 which is interpreted as undefined not zero lines... this parameter is only used in conjunction with OutOp=CLIP_ WORDWRAP
Align	This specifies the desired alignment for your text string within the specified output. If both OutRectWidth and OutRectHeight are 0 then this parameter is ignored. Valid choices are: **LEFT** - which means align text to the left in defined rectangle **CENTER** - which means center text in defined rectangle **RIGHT** - which means align text to the right in defined rectangle	0 (left)

Method	Description	Default value
OutTransColor	This is the color to set as the output file's transparent color (*only valid when output is GIF*). **Note:** Setting OutTransColor the same as BkColor will produce an image where the background is transparent, (so long as BkImage has not been specified) If BkImage has been specified, then setting OutTransColor will override any transparent color that may have been present in the background image, (i.e., the transparency in BkImage *will not* be maintained in this case). If a background image has been specified, (BkImage), and the designer has chosen a background effect of Bevel (BkEffect) with Rounded Corners, then this color will serve as the background for the rounded corner effect.	No default.
TextMarginLeft	This is the margin of pixels between the left edge of the final image and the left edge of the text graphic portion of the image when Align is LEFT. If Align is RIGHT then this will act as the left side clipping to apply to the text graphic.Note: This value is ignored by the Graphic Server if any of the following are true OutRectWidth is 0 (i.e., not specified) OutOp is CLIP_TEXTBOUND	0
TextMarginRight	This is the margin of pixels between the right edge of the final image and the right edge of the text graphic portion of the image when Align is RIGHT . If Align is LEFT then this will act as the right side clipping to apply to the text graphic. **Note:** This value is ignored by the Graphic Server if any of the following are true OutRectWidth is 0 (i.e., not specified) OutOp is CLIP_TEXTBOUND	0

Method	Description	Default value
ImgMarginLeft	Only applicable if there is a BkImage specified. This is the margin of pixels between the left edge of the final image and the left edge of the BkImage portion of the image when Align is LEFT. If Align is RIGHT, then this will act as the left side clipping to apply to the BkImage. **Note:** This value is ignored by the Graphic Server if any of the following are true OutRectWidth is 0 (i.e., not specified) OutOp is CLIP_TEXTBOUND	0
ImgMarginRight	Only applicable if there is a BkImage specified. This is the margin of pixels between the right edge of the final image and the right edge of the text graphic portion of the image when Align is RIGHT. If Align is LEFT, then this will act as the right side clipping to apply to the BkImage. **Note:** This value is ignored by the Graphic Server if any of the following are true OutRectWidth is 0 (i.e., not specified) OutOp is CLIP_TEXTBOUND	0

Graphics server functionality currently works as follows:
If the OutRectWidth and OutRectHeight are set to 0, then the OutOp defaults to NONE. However, if either or both of the OutRectWidth or OutRectHeight values are set to some value other than 0, then the OutOp defaults to CLIP_ELLIPSIS.

CLIP_WORDWRAP works as follows:
If the text string does not fit within the OutRectWidth, then the Graphics server breaks the string at a space character and adds a new line. It continues to do this for the entire text string. When the server encounters a string that does not fit but has no spaces, the server cannot break it. In this case, it will handle it as if the text will not fit, truncate the text, and append an ellipsis.

The server will also accept a limit for the number of text lines (OutMaxLines), so a designer can specify that the word wrapping will only have this many lines of text. If the OurRectHeight is specified, then OutRectHeight will take precedence if the number of lines specified will not fit.

The server will not display partial text when applying the OurRectHeight clipping, so text strings must vertically fit entirely to be included. However,

there is one exception: if the server cannot even fit one line vertically, the server will display it clipped vertically, to indicate that the server needs more space or a smaller font point size.

Example:

```
<img src=?GetImage&TextString=Millennia+is+a+Redbook
&OutRectWidth=100&OutRectHeight=80&OutOp=CLIP_WORDWRAP&OutMaxLines=3&Align
=RIGHT>
```

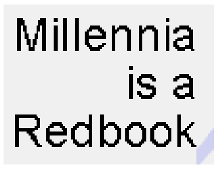


Figure 54. Example of LROutputObj

5.1.1.3 LRFontObj

Table 9 lists the font metric object, describing the font and color information for the text object.

Table 9. LRFontObj parameters

Method	Description	Default value
FontName	Name of the font to use (e.g., “Arial”). See Table 10 for available fonts to use.	Arial
FontPointSize	Desired point size to use for the text item.If the FontFitToRect value is set to true, then this field is ignored and the Image Server will compute the point size.	26
FontBold	Indicates whether font is displayed with bold attribute.	0 (not Bold)
FontItalic	Indicates whether font is displayed with italic attribute.	0 (not Italic)
FontUnderline	Indicates whether font is displayed with underline attribute.	0 (not Underline)
FontStrikeout	Indicates whether font is displayed with strikeout attribute.	0 (not Strikeout)
FontColor	The color of the text	Black (000000)

Method	Description	Default value
FontAntiAliasColor	The color for anti-aliasing text. The color selected should compliment the color of the area where the text will be presented. **Note:** If the designer selects BkColor, a solid color background, without setting OutTransColor, then FontAntiAliasColor should be the same as that solid color. Since OutTransColor was *not* specified, the resultant image will have NO transparency.If the designer sets OutTransColor (thereby indicating they wish to have a transparency), and BkColor the same color, then FontAntiAliasColor should be the same color as the background under the position where the browser will display the resultant GIF. If the designer sets BkImage, then FontAntiAliasColor should be a color that compliments the color in the image.	White (FFFFFF)
Opacity	This parameter indicates how opaque to make the text when it is combined with the background. The values are in the range of 0 to 100; where 0 would indicate that the text is totally invisible and 100 would indicate that the text is totally opaque.	100

Note: The Font Names that can be used with the Graphics server are stored in the *Resources.nsf* file, divided into four font groups: h_FontBasic, h_FontModern, h_FontOrnate and h_FontWacky.

Table 10. Fonts in Resorces.nsf

Font Group	Font	Font Group	Font
h_FontBasic	Arial	h_FontOrnate	Gallery
h_FontBasic	Arial+Black	h_FontOrnate	GardenDisplayCaps
h_FontBasic	BordeauxLight	h_FontOrnate	Ladybug
h_FontBasic	Comic+Sans+MS	h_FontOrnate	MarqueeEngraved
h_FontBasic	Comix	h_FontOrnate	Morocco
h_FontBasic	Courier+New	h_FontOrnate	Ramona
h_FontBasic	Fritz	h_FontOrnate	ReliefSerif

Font Group	Font	Font Group	Font
h_FontBasic	Georgia	h_FontOrnate	RomanStonecut
h_FontBasic	Impact	h_FontOrnate	SherwoodCaps
h_FontBasic	KabobLight	h_FontOrnate	Socrates
h_FontBasic	LitheLight	h_FontOrnate	ThorinDisplayCaps
h_FontBasic	SharnayExtraLight	h_FontOrnate	TitlingCaps
h_FontBasic	Times+New+Roman	h_FontOrnate	TypographerDisplay
h_FontBasic	Trebuchet+MS	h_FontOrnate	Vangard
h_FontBasic	UniversalLight	h_FontWacky	Aramis
h_FontBasic	Verdana	h_FontWacky	Carnivale
h_FontModern	AireoDisplay	h_FontWacky	Cut-n-Paste
h_FontModern	Arcane	h_FontWacky	Duo-Line
h_FontModern	ArcaneBroad	h_FontWacky	Dustine
h_FontModern	BordeauxBlackOutlin	h_FontWacky	FastracFashion
h_FontModern	ClearlyGothicLight	h_FontWacky	Funstuff
h_FontModern	DebevicCircular	h_FontWacky	Gecko
h_FontModern	Encino	h_FontWacky	Gymnastics
h_FontModern	EyechartCondensed	h_FontWacky	HotTamale
h_FontModern	FuturistExtrabold	h_FontWacky	Mandrel
h_FontModern	Gatsby	h_FontWacky	MandrelOutline
h_FontModern	Kensington	h_FontWacky	Maraca+Extras
h_FontModern	ModaerneLight	h_FontWacky	MarqueeFlash
h_FontModern	Modern	h_FontWacky	NeonCaps
h_FontModern	NouveauAsta	h_FontWacky	OrientNarrow
h_FontModern	Sadelle	h_FontWacky	Promenade
h_FontModern	VagabondShadow	h_FontWacky	PyxidCondensed
h_FontModern	WeekendInParis	h_FontWacky	QuarkNeon
h_FontModern	XpressiveLight	h_FontWacky	SaminoaDisplay

Font Group	Font	Font Group	Font
h_FontOrnate	Arruba	h_FontWacky	Shower
h_FontOrnate	CruiselineDisplayCa	h_FontWacky	TabathaFresco
h_FontOrnate	Currency	h_FontWacky	ToolShopCaps
h_FontOrnate	CurrencyOutline	h_FontWacky	Truffle
h_FontOrnate	DesignerTitlingCaps	h_FontWacky	Trufflette
h_FontOrnate	DropCaps	h_FontWacky	Webdings

Example:

```
<img src=?GetImage&TextString=Millennia+Redbook&FontName=ModaerneLight
&FontPointSize=24&FontColor=ff0000>
```

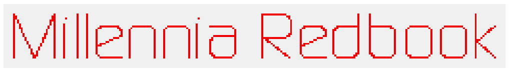


Figure 55. Example of LRFontObj with font ModaerneLight

```
<img src=?GetImage&TextString=Millennia+Redbook&FontName=Cut-n-Paste
&FontPointSize=24>
```

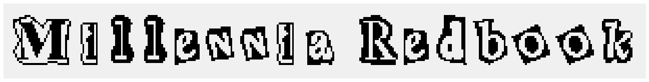


Figure 56. Example of LRFontObj with font Cut-n-Paste

5.1.1.4 LRFileFormat

Table 11 provides the output file description, which applies to the entire image.

Table 11. LRFileFormat parameters

Method	Description	Default value
FileFormat	Desired format for the output file, either JPEG or GIF.	GIF

Example:

```
<img src=?GetImage&TextString=Millennia+Redbook&FileFormat=JPEG>
```

5.1.1.5 LRCompressInfo

Table 12 lists the compression information. Currently used only for writing JPEG compressed images, this applies to the entire image.

Table 12. LRCompressInfo parameters

Method	Description	Default value
JpegQuality	Quality factor for the JPEG compression, in a range from 5 – 100. The higher the value will give higher image quality, higher file size. The lower the value will give lower quality image, lower file size.	85
JpegProgressive	If true, then the image will be compressed as it progresses. This gives the capability to viewers of displaying a JPEG compressed image in portions, but requires more time to compress and decompress.	0 (FALSE)

Example:

```
<img src=?GetImage&TextString=Millennia+Redbook&FileFormat=JPEG
&JpegQuality=50&JpegProgressive=1>
```

5.1.1.6 LREffectObj

Table 13 lists and describes the special effects object.

Table 13. LREffectObj parameters

Method	Description	Default value
EffectType	The type of special effect to apply. **None** **Blur** **Shadow**	None

Example

```
<img src=?GetImage&TextString=Millennia+Redbook&EffectType=Shadow>
```

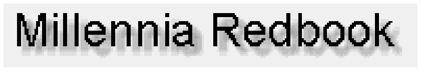


Figure 57. Example of LREffectObj

5.1.1.7 LREffectParams

Table 14 lists and describes the parameters for special effects.

Table 14. LREffectParams parameters

Method	Description	Default value
If the effect type is **Blur:**		
BlurFactor	Amount of blur, (how blurry to make the image)	5
If the effect type is **Shadow:**		
BlurFactor	Amount of blur of shadow, (how blurry to make the shadow)	5
ShadowColor	Color of the shadow	Gray (808080)
ShadowXOffset	The horizontal offset of the shadow, positive indicates that the shadow falls to the right of the image, negative indicates that the shadow falls to the left of the image.	4 pixels
ShadowYOffset	The vertical offset of the shadow, positive indicates that the shadow falls to below the image, negative indicates that the shadow falls above the image.	4 pixels

Example

```
<img src=?GetImage&TextString=Millennia+Redbook&EffectType=Shadow
&ShadowColor=ff0000&ShadowXOffset=10&ShadowYOffset=10>
```

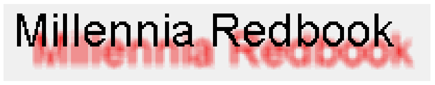


Figure 58. Example of LREffectParams

5.1.1.8 LRBackground

Table 15 lists and describes the parameters for the Background Layer.

Table 15. LRBackground parameters

Method	Description	Default value
BkColor	The solid color to set for the background. If BkImage has been specified, then this parameter is ignored.	White (ffffff)

Method	**Description**	**Default value**
BkImage	This must be a Domino Command URL to the image to use as a background. This image MUST be an attachment to a Notes Database. **Notes:** This *must* be a valid fully qualified or relative URL to a GIF. If the designer has *not* specified values for OutRectWidth and OutRectHeight, then the resultant image size for the output is that of the background image rectangle. The image will be clipped to this rectangle and any text string that falls outside this rectangle will not appear in the resultant composite image. Otherwise, the rules when OutRectWidth and OutRectHeight are set to values other than 0 apply.If this image has a transparency and OutTransColor has *not* been specified, then this transparent color will be maintained so that the output GIF has the same transparent color.	No default
BkImageAlt	This must be a Domino Command URL to the alternate image to use. See rules for BkImage. This image will be used under the following circumstances *only* OutOp = CLIP_WORDWRAP *and* when the text graphic results in more than one line. This would allow for designers to pick background images that are suitable when the Graphics Server wraps text. If no BkImageAlt is specified, then it will just go with BkImage.	No default
BkEffect	This parameter indicates the effect, if any, to apply to the BkImage if any. Valid entries for this parameter are: **NONE** - no effect **BEVEL** - bevel the edge of the BkImage **BEVELHILITE** - same as bevel but the beveled edge of the image will be highlighted, (to indicate mouse over, etc.). The highlighting is done by taking the BkColor value and choosing the opposite value and using that in the bevel.	NONE

Method	Description	Default value
BkMapColor	This is a color mapping parameter, that allows the designer to map one color to another. This is a string with the following format: rrggbb,rrggbb where the first 6 characters are the "From Color" and then the next character is the delimiter, then the last 6 characters are the "To Color". When this parameter is present and a BkImage has been specified then The Graphics Server will convert pixels in this image that have the value of the "FromColor" into the "ToColor". Note that this mapping is performed prior to other operations.	No default
BkImgAlign	This is the horizontal alignment of the background with respect to the output rectangle. Allowed values are: **CENTER** - center the background image between the left and right. **LEFT** - left-align background image. **RIGHT** - right-align background image.	LEFT
BkImgVAlign	This is the vertical alignment of the background with respect to the output rectangle. Allowed values are **CENTER** - center the background image between the top and bottom. **TOP** - top-align background image. **BOTTOM** - bottom-align background image.	CENTER

Example

```
<img src=?GetImage&TextString=Millennia
&BkImage=/quickplace/redbook/Main.nsf/$All/Limerick/$FILE/m.gif?OpenElemen
t&OutRectWidth=100&OutRectHeight=50&BkImgAlign=RIGHT&BkImgVAlign=TOP>
```

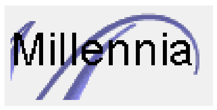


Figure 59. Example of LRBackground

5.1.1.9 Bevel

Table 16 lists and describes the parameters for the Bevel Effect on the Background Image.

Table 16. Bevel parameters

Method	Description	Default value
BevelSize	The size of bevel to apply to the image, in pixels.	8
BevelLightSource	The light effect to apply to the bevel. Valid parameters are: **UPPER_LEFT** - the light comes from the upper left. **UPPER_RIGHT** - the light comes from the upper right **LOWER_LEFT** - the light comes from the lower left **LOWER_RIGHT** - the light comes from the lower right **Note:** The light and dark edges of the bevel depend on the selection for BevelType. Consider an example; suppose the designer selects UPPER_LEFT and SIMPLE_OUTER. The the top and left bevels will be "light" while the right and bottom bevels will be "dark". If the designer chose SIMPLE_INNER instead then the opposite would be true, the top and left bevels will be "dark" while the right and bottom bevels will be "light".	UPPER_LEFT
BevelType	This indicates type of bevel; valid entries are: **SIMPLE_OUTER** - this is the simple bevel with an outer bevel (indicating that the image is raised). **SIMPLE_INNER** - this is the simple bevel with an inner bevel (indicating that the image is depressed). **Note:** If a designer wishes to generate a button with this effect, they could create two bevelled images, one with outer bevel and one with inner to create the animation action of buttons being pressed and released.	SIMPLE_OUTER

Method	Description	Default value
BevelCorner	This parameter indicates how to treat the corners of the bevelled image. Valid entries are **SQUARE** - the image corners are left square **ROUNDED** - the image corners are rounded	SQUARE

Example

```
<img src=?GetImage&TextString=Millennia
&BkImage=/quickplace/redbook/Main.nsf/$All/Limerick/$FILE/m.gif?OpenElemen
t&OutRectWidth=100&OutRectHeight=50&BkImgAlign=RIGHT&BkImgVAlign=TOP&BkEff
ect=BEVEL&BevelSize=10&BevelCorner=ROUNDED>
```

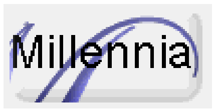


Figure 60. Example of Bevel parameters

5.1.1.10 LRAnimation

Table 17 lists and describes the parameter which specifies the animation, if any.

Table 17. LRAnimation parameters

Method	Description	Default value
Animation	Specifies the animation, acceptable values are **SLIDEIN** - the text image will slide into the image from some specified direction. **SLIDEOUT** - the text image will slide out of the image to some specified direction. **SLIDEINOUT** - the text image will slide into the image from some specified direction, then slide out to some other specified direction. These directions can be different. **FADEIN** - the text image will fade into the image. **FADEOUT** - the text image will fade out, leaving only the background. **UNDULATION** - similar to fade in/out, the text graphic will color shift from the text color to the other color specified in AnExtra	No default
AnExtra	For the SLIDEIN, SLIDEOUT and SLIDEINOUT effects, the designer can set the direction. For SLIDEIN and SLIDEOUT a single direction is required, while SLIDEINOUT requires two directions, "slide in from where" and "out to where". When indicating the dual direction value, the designer should input the string using the format: "FromDirection,ToDirection" Acceptable directions are **LEFT** **RIGHT** **TOP** **BOTTOM** For UNDULATION effects, this is the color to shift to specified as rgb triplet.	If a slide effect is not chosen then this has no default. If effect is SLIDEIN or SLIDEOUT, then this defaults to LEFT. If the effect is SLIDEINOUT then the default is LEFT,LEFT. If the effect is SLIDEINOUT and only one direction was entered, then the other defaults to the same direction. If the effect is UNDULATION then the default is black (000000).

Method	Description	Default value
AnFrames	This indicates the number of frames desired for the animation, more frames make smoother animation, but each frame adds to the image size. You should get by on as few frames as possible.	10
AnDelay	This indicates the amount to time to delay between frames of the animation, in 1/100 seconds, (i.e. if AnDelay=100 then we delay between frames for 1 second).	20
AnLoop	This indicates whether or not the animation should occur one time or in a loop. 0 indicates a one time animation 1 indicates a loop	1
AnIterations	If AnLoop is 1, then this indicates how many times to loop. A value of 0 indicates and infinite looping. **NOTE:** not all browsers recognize this item, and default to endless loops	0

Example

```
<img src=?GetImage&TextString=Millennia+Redbook&FontColor=ff0000
&Animation=UNDULATION&AnExtra=0000ff&AnFrames=20>
```

5.1.2 Method: Merge

The merge method is used to combine two images. It takes a BaseImage object and an OverlayImage object. The OverlayImage is placed on top of the BaseImage to create the merged image result. The BaseImage *must* be equal to or larger in width and height than the OverlayImage.

5.1.2.1 LRBaseImage

Table 18 lists and describes the base image to use for making a combined image.

Table 18. LRBaseImage parameters

Method	Description	Default value
BaseImage	This is a Domino Command URL which points to the image to use as the base image for creating the merged image. It must be of format JPG or GIF.	No default

Method	Description	Default value
BaseMapColor	This is a color mapping parameter, that allows the designer to map one color to another. This is a string with the following format: rrggbb,rrggbb where the first 6 characters are the "From Color" and then the next character is the delimiter, then the last 6 characters are the "To Color". When this parameter is present then the Graphics Server will convert pixels in this image that have the value of the "FromColor" into the "ToColor". Note that this mapping is performed prior to other operations.	No default

Example

See 5.1.0.1, “LRMethod” on page 112.

5.1.2.2 LROverlayImage

Table 19 lists and describes the overlay image to use for making combined image. This image will be placed on top of the base image.

Table 19. LROverlayImage parameters

Method	Description	Default value
OverlayImage	This is a Domino Command URL which points to the image to use as the overlay image for creating the merged image. It must be of format JPG or GIF.	No default
OverlayMapColor	This is a color mapping parameter, that allows the designer to map one color to another. This is a string with the following format: rrggbb,rrggbb where the first 6 characters are the "From Color" and then the next character is the delimiter, then the last 6 characters are the "To Color". When this parameter is present then the Graphics Server will convert pixels in this image that have the value of the "FromColor" into the "ToColor". Note that this mapping is performed prior to other operations.	No default

Example

See 5.1.0.1, “LRMethod” on page 112

5.1.2.3 LRPosition

Table 20 lists and describes the position information to use when placing the overlay image on the base image.

Table 20. LRPosition parameters

Method	Description	Default value
Align	Indicates how to horizontally position the overlay image on the base image. Acceptable values are: **CENTER** - place the overlay image on the base image such that the horizontal center point of the overlay image coincides with the horizontal center position of the base image. **LEFT** - place the overlay image on the base image such that the left edge of the the overlay image coincides with the left edge of the base image. **RIGHT** - place the overlay image on the base image such that the right edge of the the overlay image coincides with the right edge of the base image. This value is ignored if XOrigin is specified.	CENTER
VAlign	Indicates how to vertically position the overlay image on the base image. Acceptable values are **CENTER** - place the overlay image on the base image such that the vertical center point of the overlay image coincides with the vertical center position of the base image. **TOP** - place the overlay image on the base image such that the top edge of the the overlay image coincides with the top edge of the base image. **BOTTOM** - place the overlay image on the base image such that the bottom edge of the overlay image coincides with the bottom edge of the base image. This value is ignored if YOrigin is specified.	CENTER

Method	Description	Default value
XOrigin	The X coordinate value to use when placing the overlay image on the base image. This value is relative to the upper left of the base image, and is in units of pixels. If this value is specified, then we ignore the value for Align.	
YOrigin	The Y coordinate value to use when placing the overlay image on the base image. This value is relative to the upper left of the base image, and is in units of pixels. If this value is specified, then we ignore the value for VAlign.	
Opacity	This parameter indicates how opaque to make the overlay image when it is combined with the base image. The values are in the range of 0 to 100; where 0 would indicate that the overlay is totally invisible and 100 would indicate that the overlay is totally opaque. Note: When setting an Opacity of less than 100, the OverlayImage GIF MUST NOT have a transparent color. If, however, the Opacity equals 100 (it is totally opaque), then the OverlayImage GIF can have a transparent color.	100

Example

```
<img src=?GetImage&Method=MergeImage
&BaseImage=/quickplace/redbook/Main.nsf/$All/Limerick/$FILE/m.gif?OpenElem
ent&OverlayImage=/quickplace/redbook/Main.nsf/$All/Limerick/$FILE/RedTrans
p.gif?OpenElement&Opacity=50>
```

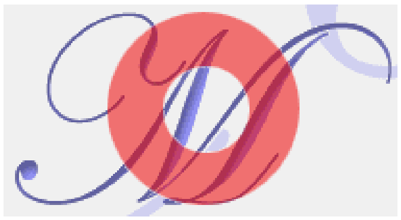


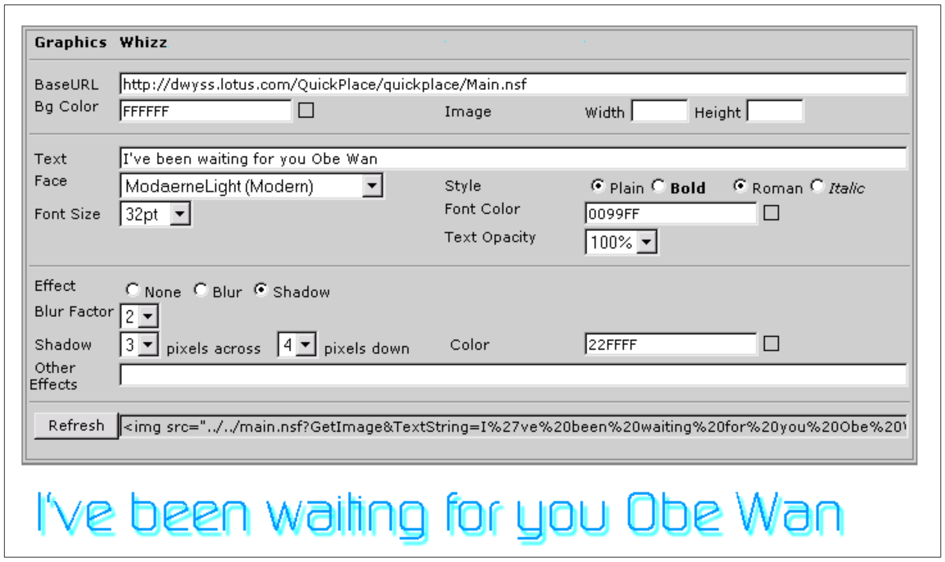
Figure 61. Example of LRPosition

5.1.3 Graphics Server Image Wizard: GraphicsWhizz

The following section describes the GraphicsWhizz, an HTML page which creates images using the Graphics Server; seeFigure 62.

The GraphicsWhizz allows you to set the parameters to create text effects using drop-down lists, radio buttons, and input fields. It creates a URL that you can paste straight into your Theme layout and any other HTML files used in QuickPlace. It is also a handy tool for learning how to create images using the Graphics server because you can watch the URL being created in the output field on the fly.

GraphicsWhizz does not use all of the parameters available via the graphics server, but it does use the most commonly used ones. Once you see how it works, you will be able to start creating your own images using the same syntax. It gives you the ability to merge automatically created Parameters with parameters you type into the Other Effects field.


Graphics Whizz

BaseURL: http://dwyss.lotus.com/QuickPlace/quickplace/Main.nsf
Bg Color: FFFFFF   Image Width: Height:

Text: I've been waiting for you Obe Wan
Face: ModaerneLight (Modern)   Style: [✓ Plain] Bold [✓ Roman] Italic
Font Size: 32pt   Font Color: 0099FF
Text Opacity: 100%

Effect: None Blur [✓ Shadow]
Blur Factor: 2
Shadow: 3 pixels across 4 pixels down   Color: 22FFFF
Other Effects:

Refresh   <img src="../../main.nsf?GetImage&TextString=I%27ve%20been%20waiting%20for%20you%20Obe%20

Figure 62. The GraphicsWhizz

The following section details a few of the key functions in the GrahpicsWhizz HTML page.

The main functionality of the page is in the fnLimerickerizer function. It gets its name from the former name of the QuickPlace Graphics Server: Limerick. It creates a URL to reset the SRC of an image at the bottom of the screen.

It sets a number of variables using the fields in the form, and builds the URL; Table 21 lists the parts. You can use the same syntax to make hardcoded URLs.

Table 21. Anatomy of a QuickPlace Theme

Parameter	Example Value	Description
URL	http:// .... /main.nsf	The Main.nsf address of a QuickPlace Server you have access to.
?GetImage		Call the Graphics Server. This parameter does not accept a value.
&TextString=	I've been waiting ...	The string to display. You should escape apostrophes with "%27" and spaces with either a Plus or "%20". In the GraphicsWhizz this is done automatically.
&FontName=	ModaerneLight	The font to display. Choose only fonts on the server if you want all users to be able to use this font.
&FontPointSize=	32	Size of the font. Remember that large fonts will create larger images and therefore larger file sizes. The Graphics Whizz only allows you a few choices, but this is only limited by good taste.
&FontBold=	0	Sets the font as being plain.
&`FontItalic`	1	Sets the font as being italic.
&fontColor=	0099FF	Blue color in hex format.
&shadowColor=	22FFFF	Light blue in hex format color.
&`EffectType=`	Shadow	Chooses a shadow effect.
&blurFactor=	2	Blurs the image by 2 pixels.
&ShadowXOffset=	3	Pushes the shadow to the right by 3 pixels
&ShadowYOffset=	4	Pushes the shadow to the left by 4 pixels.
&BkColor=	FFFFFF	Sets a white background in hex color.

The GraphicsWhizz.htm file is available for download from the IBM Redbooks Web site. See Appendix L, "Additional Web material" on page 425 for instructions on how to get the sample.

5.2 Summary

In this chapter we have explained the different parameters you can use to have the QuickPlace Graphic server create graphics images containing dynamic text strings. We have also given you a reference to a tool that makes it very easy to create the image creation URL command.

Chapter 6. Forms in QuickPlace

This chapter explains how to define your own forms to create pages within QuickPlace.

QuickPlace provides you with a lot of ways to add new content to your QuickPlace. These include options to upload a document and send a notification, add a meeting to the calendar, or add a task into the QuickPlace. By clicking **New...**, the user gets a list of forms included in QuickPlace that can be used to add a new document to it.

The forms provided are sufficient for many uses, but do not give you any task-specific ways of adding content to the QuickPlace. To do this, you have to create your own form and adapt it to your particular needs.

There are three ways to create forms:

- Create a form using standard QuickPlace fields
- Import a form created in Microsoft Office
- Import a form created in an HTML editor

We discuss each process of creating a form in this chapter, and explain how to use some QuickPlace-specific components and JavaScript functions in your forms.

When creating a form, you can adapt it to the workflow in your team; this is also explained in this chapter.

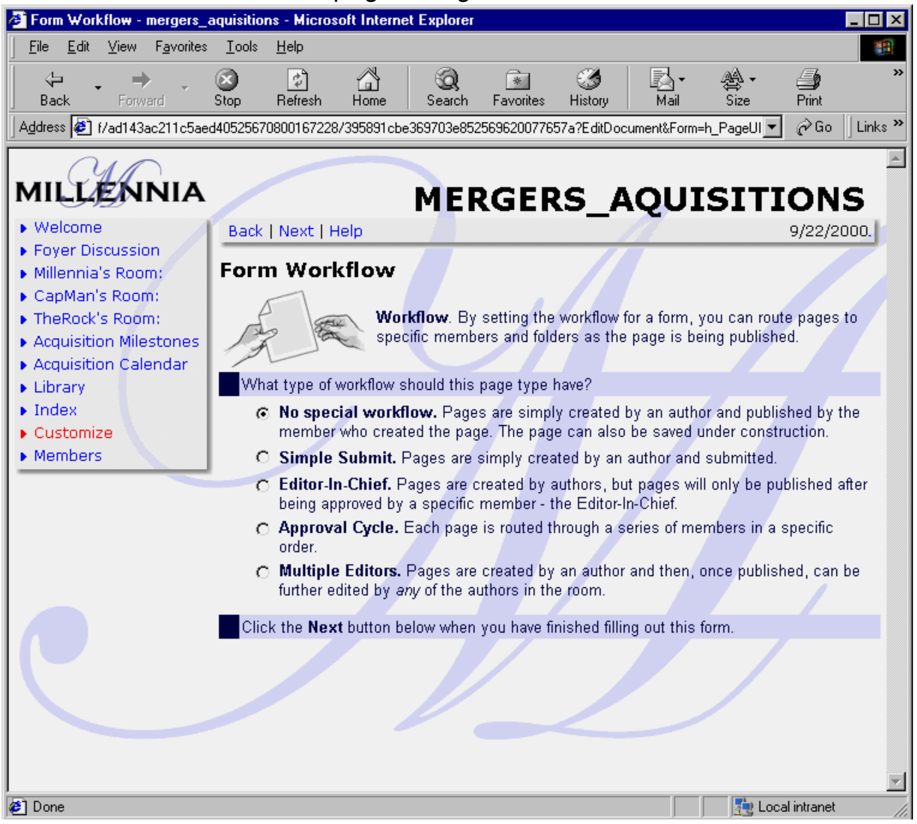
6.1 Defining a type of workflow

To accomplish your virtual teams goals, you have to go through certain tasks, fullfilling a number of activities in a certain order, and do this within a given time frame. By defining the way your team's or company's goals are accomplished, you define the workflow in your organization. You can adapt QuickPlace to reflect your workflow, your way of doing business.

Before you add a type of workflow to a form, decide who is going to use this form and what should happen to the page once it has been created by an author. If you plan to use workflow in QuickPlace, try not to devise really complicated schemes, as the types of workflow offered in QuickPlace reflect mainly the basic ways to conduct workflow. If you need to portray more complicated forms of workflow, consider connecting your QuickPlace to a

Domino Workflow environment, as described in Appendix 10.4, “Integrating with Domino Workflow” on page 299.

You can see the workflow page in Figure 63.


Form Workflow - mergers_aquisitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print
Address f/ad143ac211c5aed40525670800167228/395891cbe369703e852569620077657a?EditDocument&Form=h_PageUI
Go Links
MILLENNIA
Welcome
Foyer Discussion
Millennia's Room:
CapMan's Room:
TheRock's Room:
Acquisition Milestones
Acquisition Calendar
Library
Index
Customize
Members
MERGERS_AQUISITIONS
Back | Next | Help
9/22/2000.
Done
Local intranet

Form Workflow

Workflow. By setting the workflow for a form, you can route pages to specific members and folders as the page is being published.

What type of workflow should this page type have?

[✓ **No special workflow.** Pages are simply created by an author and published by the member who created the page. The page can also be saved under construction.]

Simple Submit. Pages are simply created by an author and submitted.

Editor-In-Chief. Pages are created by authors, but pages will only be published after being approved by a specific member - the Editor-In-Chief.

Approval Cycle. Each page is routed through a series of members in a specific order.

Multiple Editors. Pages are created by an author and then, once published, can be further edited by *any* of the authors in the room.

Click the **Next** button below when you have finished filling out this form.

Figure 63. The Form Workflow page

QuickPlace lets you integrate a form into your organization’s workflow. Table 22 shows the options that are available on the Form Workflow page.

Table 22. Short description of workflow in QuickPlace

Workflow type	**Description**
No special workflow	Pages are simply created by an author and published by the member who created the page.
Simple Submit	Pages are simply created by an author and submitted. This will add an submit button to the button bar at the top of the page.

Workflow type	Description
Editor In Chief	Pages are created by authors, but pages will only be published after being approved by a specific member - the Editor-In-Chief.
Approval Cycle	Each page is routed through a series of members in a specific order.
Multiple Forms	Pages are created by an author and then, once published, can be further edited by *any* of the authors in the room.

In the following sections, we provide more detail about these options.

No special workflow

This option will allow members to publish their pages based on this form, without first getting approval from any other member of the QuickPlace, either as a draft or in their final form.

Simple Submit

Choose this option if the pages created by this form don't have to undergo review, the users should not be able to save pages created by this form as drafts, and you want to be able to rename the **Publish** button.

When an author creates a page using this form, the following events happen. (We assume here that the Simple Submit workflow was used to rename the **Publish** button to the **Post to Project Milestones** button, and the form workflow set to publish to that folder):

1. The author creates the page and clicks **Post to Project Milestones**.
2. The page is published to the **Project Milestones** folder.

Editor in Chief

Select this option if you want a single member (called Editor in Chief) to review each page created with this form.

When an author creates a page that has to be reviewed by an Editor in Chief, the following events occur:

1. The author finishes editing the page and notifies the Editor in Chief that the page is ready for review by clicking the **Submit** button. Technically, the author has passed the right to edit the page to the Editor in Chief.
   a. Upon submitting the page, the author can choose to send the Editor in Chief a note about the page by e-mail. The note only appears in the e-mail message, not on the page itself.

   b. The author can also save the page under construction to continue editing at a later time before submitting it to the Editor in Chief.

2. The Editor in Chief receives an e-mail message containing the author's note (if one has been created) with a link to the new page.

3. The Editor in Chief reviews the pages and does one of the following:

   a. Edits the page (if necessary) and, by publishing, approves it. The Editor in Chief can select to notify the author that the page has been published.

   b. Rejects the page. The Editor in Chief can choose to attach a note to the rejection message.

The author receives an e-mail message saying that either the page has been published or rejected. If the page has been rejected, the author can revise the page and then resubmit it to the Editor in Chief for approval.

Approval Cycle
Choose this type of workflow if you want more than one member of your QuickPlace to review pages created with the form. This is similar to the Editor in Chief process, except that it includes more that one reviewer.

After choosing Approval Cycle, you define which members, in which order, should review the document. You can also set restrictions on who may read or edit the final page, and set the member who is responsible for editing the page when it is rejected.

If you define an approval cycle with two reviewers, and set the folder Project Milestones as the destination for final approved pages, the following events occur:

1. The author finishes editing the page and notifies the first reviewer that it can be reviewed by clicking the **Submit** button.

   a. The author can choose to send a note on the page in an e-mail message to the first reviewer. This note only appears in the e-mail message, not the page itself.

   b. The author can also choose to save the page under construction to continue editing it at a later time.

2. The first reviewer receives an e-mail notifying him that the page is ready for review. It contains the note from the author (if one was created) and a link to the page awaiting approval. The first reviewer reviews the page and takes one of the two following actions:

   a. Reads and edits the page (if necessary) and submits it to the next reviewer.
   b. Rejects the page. The reviewer can choose to create a note to accompany the rejection notice.
3. If the first reviewer rejects the page, the author receives an e-mail notifying him of the rejection. He then can change the page and resubmit it to the first reviewer, repeating the first two steps of this procedure.
4. If the first reviewer approves the page, the second reviewer receives an e-mail with a link to the page awaiting approval and a note from the first reviewer (if one was created). The reviewer takes one of the following actions:
   a. Reviews, edits (if necessary) and approves the page. The page is then published in the Project Milestones folder.
   b. Rejects the page. The author receives an e-mail message stating that the page has been rejected. The second reviewer can choose to create a note to accompany the notification.
5. The author receives the notification, corrects the page and resubmits it to the second reviewer, who in turn can then either reject it again or publish it.

Multiple editors
Choose this option if you want to grant all authors in the QuickPlace edit rights to the page created by this form.

This is useful if you have a document that has to be viewed by all and you want all authors to add their thoughts to it.

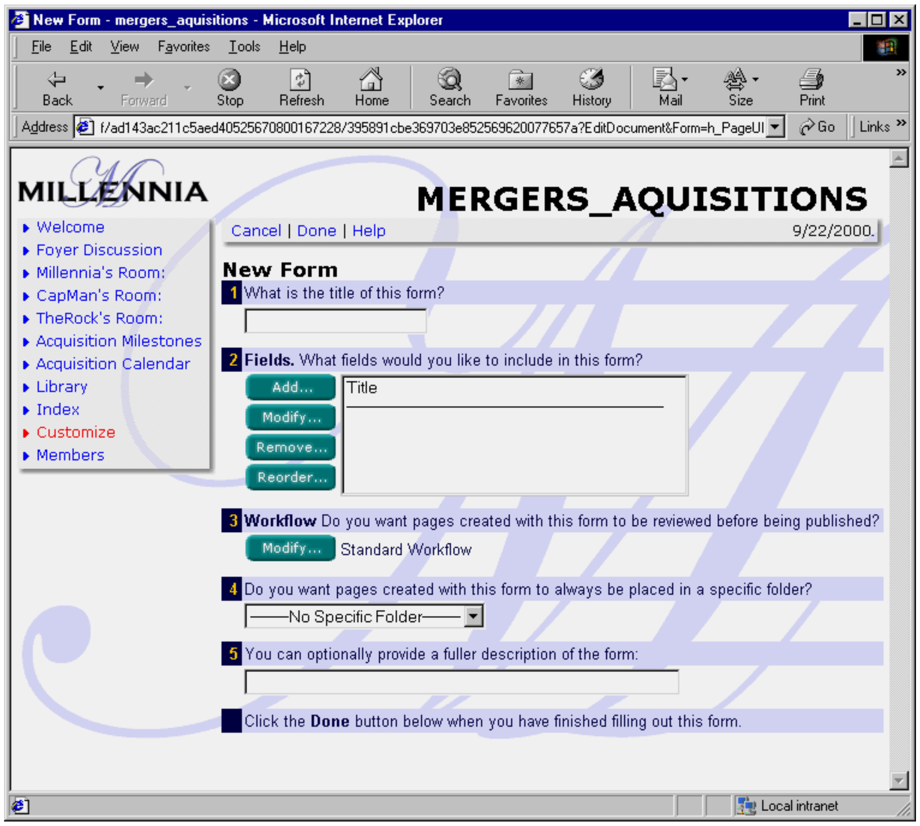
We used this type on the poll form that we discuss in 6.4.2, as the user actually edits the page when he or she casts a vote in the poll. Thus, the member has to be an author to take part in the voting.

6.2 Creating a Form within QuickPlace

Creating a Form within QuickPlace is a simple process. If you need to generate a simple Form with just a few fields in it, use the feature within QuickPlace. If you need a Form that is more sophisticated, or have to include JavaScript to do checks on fields, for example, you will have to create the Form outside of QuickPlace and import it.

Imagine you want the users to be able to add events to the calendar when the page is published, and also have the pages published in a specific folder.

To do this, choose **Customize** from the main menu, and click **New Form**. On the next screen, choose **Simple Form** and click **Next**. Figure 64 shows the Edit Form page.



MERGERS_AQUISITIONS

9/22/2000.

New Form

1 What is the title of this form?

2 **Fields.** What fields would you like to include in this form?

Title

3 **Workflow** Do you want pages created with this form to be reviewed before being published?

Standard Workflow

4 Do you want pages created with this form to always be placed in a specific folder?

——No Specific Folder——

5 You can optionally provide a fuller description of the form:

Click the **Done** button below when you have finished filling out this form.

Figure 64. The New Form page

Add fields to the Form by clicking **Add**, and select a field. For example, to add the page created by this form to the calendar automatically, choose the field **Event date and time**.

QuickPlace gives you a choice of standard field types you can use to create your Form. The types are shown in Table 23.

Table 23. QuickPlace Form field types

Field type	**Function**
Plain Text	Presents a one-line unformatted text field
Text Area	Presents a multi-line unformatted text field

Field type	Function
Pop-up list	Presents a list of choices from which to choose
Time Pop-up	Presents a time pop-up
Name Pop-up	Presents a list of QuickPlace members
Attachments	Presents a field for adding file attachments
Rich Text	Presents a field where the author can enter formatted text and images
Calendar Date-Time	Presents a combination of date and time fields so that pages created with this form will automatically be added to this QuickPlace's Calendar
Task	Presents a combination of task-related fields so that pages created with this form can be tracked as tasks

Furthermore, there are a number of fields that are non-editable, but provide additional information on the form. They are shown in Table 24.

Table 24. Special non-editable fields for QuickPlace Forms

Field type	Function
Notification Indicator	Used to automatically send e-mail to individuals
Non-Editable Rich Text	Presents non-editable text and graphics on the form. Typically this is used to provide an attractive banner on top of the form
Page Author	Presents the non-editable name of the author who created the page with this form
Creation Date	Presents the non-editable date that the page was created
Last Modified Date	Presents the non-editable last modification date of the page
Page Size	Presents the non-editable size of the page
Serial Number	Presents a non-editable unique identifier for each page created with the form (for example, Purchase Order Number). You cannot change the format of this field. **Note:** The name *serial* is a bit misleading. This field creates an assured unique identifier through a combination of the date, time and the users identity. It is not a number that simply is incremented

Add the fields you need for your form by following the instructions on the screen.

If you want the pages created by this form to be published in a specific folder, choose this folder from the drop-down list.

As an option, you can also give a fuller description of what the form does. This description appears next to the name of the form when the user clicks **New... .**

Setting tasks

Besides choosing a type of workflow for forms created in a QuickPlace, you can also choose to have task settings on the pages that are created using the form and add them to the Tasks list in your QuickPlace. To do this, add the field **Task** from the Add Field page and click **Next**. The following page lets you name the task field, and set a start date and the priority for the task. You can also set the initial choice for the priority.

When you are finished adding fields to the form, click **Done** to save the form. Authors can create pages based on the form by clicking **New...** in the button bar and selecting the form from the following page.

6.2.1 How to access field values from PlaceBots

In order to avoid conflicts with system fields and be able to handle the names internally in the code, QuickPlace adds the prefix *c_* to the field name you specified and strips away characters it cannot handle, such as blank spaces.

If you want to access documents created with a QuickPlace form from a PlaceBot, you need to know the internal name of your field (see Chapter 7). For simple field names, you can simply deduct the internal name from the name entered during design of the form. See the first two lines in Table 25 for examples of this.

However, if you use characters other than space, 0-9 and a-Z in your field names, you should create a test document with your form. Open the document in your browser and choose to view the source. In the HTML source, you can find the field and see the internal field name used by QuickPlace. The third row in Table 25 shows an example where several characters have been stripped away.

Table 25. How QuickPlace names simple form fields internally

Field name entered during design	**Internal field name**
Name	c_Name
Merchant ID	c_MerchantID
Do you know your ÆØÅ? (optional)	c_Doyouknowyouroptional

Note: QuickPlace does not change field names on forms created in MS Office or HTML so in those cases you should prefix your field names with *c_* to avoid conflicts with internal QuickPlace fields.

6.3 Upload a form created in MS Office

If the form requires some fields that are not part of the QuickPlace form creation page, you can create your own form either in MS Office or in an HTML editor.

Creating a form in MS Office is fairly easy and quickly done. If you need a standard form that includes field QuickPlace does not provide, or if you don't know much HTML, creating a form in MS Office is the right choice for you.

We will now describe the steps needed to create and import a MS Office form into QuickPlace. We use MS Word as an example.

Create a new Word document, and add Web form fields by clicking the field icon in the Web tools toolbar (see Figure 65). Save the document as a Word file when you are finished. QuickPlace will automatically convert the document to HTML when it is being imported.

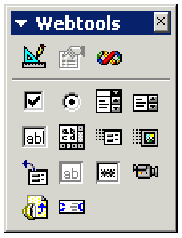


Figure 65. MS Word Web tools toolbar

Standard HTML form fields

The Web tools toolbar in Word offers eleven standard fields to use in a document:

- Checkbox control
- Option or Radio button control
- Drop Down box control
- Listbox control
- Textbox control
- Text Area control
- Submit control
- Submit with Image control
- Reset control

- Hidden control
- Password control

By clicking **Properties**, you can assign an HTML name and a value to the field.

ActiveX controls

You may also add ActiveX controls, but keep in mind that these are only usable for Internet Explorer users.

ActiveX controls are added from the Control toolbox. Again, add a control by clicking the icon in the toolbox, and set the properties by clicking **Properties**.

The following ActiveX controls are available from the tool box:

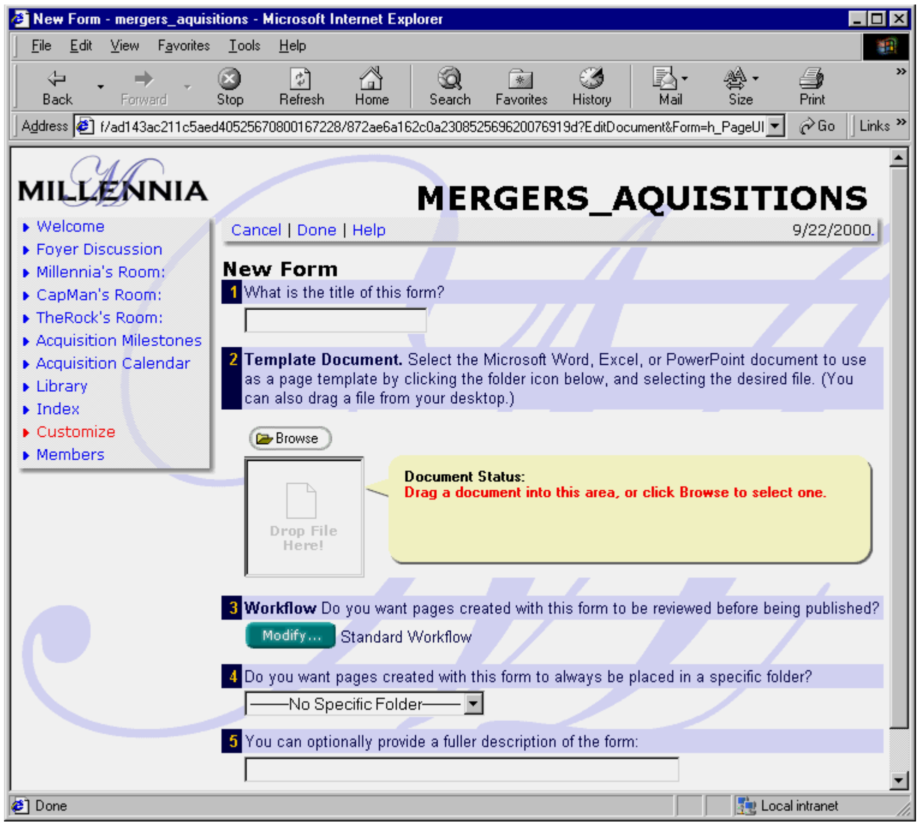
- Check box
- Spin button
- Scroll bar
- Label
- Text box
- Command button
- Option or Radio button
- List box
- Combo box
- Toggle button
- Image

You can register additional ActiveX controls by clicking **More Controls** and registering the ones you need.

For Details on setting up a form in MS Word, refer to the online help and search for **Create a form for the Web**.

You do not have to include a submit and cancel button in your form, as these can be provided by QuickPlace. When you are finished editing your document, save it as a MS Word document and upload it to the QuickPlace.

To upload, click **Customize** and select **Form**. On the next page, click **New form**, and select **Microsoft Office form**, then click **Next**.


New Form - mergers_aquisitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print
Address f/ad143ac211c5aed40525670800167228/872ae6a162c0a230852569620076919d?EditDocument&Form=h_PageUI Go Links
MILLENNIA
MERGERS_AQUISITIONS
Welcome
Foyer Discussion
Millennia's Room:
CapMan's Room:
TheRock's Room:
Acquisition Milestones
Acquisition Calendar
Library
Index
Customize
Members
Cancel | Done | Help
9/22/2000.
Browse
Drop File Here!
Modify...
Done
Local intranet

New Form

1 What is the title of this form?

2 **Template Document.** Select the Microsoft Word, Excel, or PowerPoint document to use as a page template by clicking the folder icon below, and selecting the desired file. (You can also drag a file from your desktop.)

Document Status:
Drag a document into this area, or click Browse to select one.

3 **Workflow** Do you want pages created with this form to be reviewed before being published?

Standard Workflow

4 Do you want pages created with this form to always be placed in a specific folder?

——No Specific Folder——

5 You can optionally provide a fuller description of the form:

Figure 66. The upload MS Office Form page

Fill in the fields on the next page, which is shown in Figure 66. You can drag and drop your MS Word document into the input area (also called a *bucket*) or select it by clicking **Browse**.

You can choose a workflow option from the list, as described earlier in this chapter.

To change the standard **Publish** button for your Form, select **Workflow** and choose **Simple submit**. This will add a submit button into the button bar at the top of your Form when it is filled in. Click **Next** and provide a name for the submit control on the next page. Click **Next**.

Select the folder you want the pages to be published in from the drop-down list. You can add a short description of your Form, if you choose.

Click **Done** when you are finished. The Form will be published to the QuickPlace.

Authors can publish pages by selecting **New...** from the button bar and selecting your Form from the list.

6.4 Upload a manually created HTML Form

If you need more control over how your Form appears in the QuickPlace, or need to include JavaScript, you have to create a Form manually in an HTML editor. A good knowledge of HTML and JavaScript, if needed, is required to do this. Refer to an HTML or JavaScript reference to learn more.

We have created two examples to show how to create Forms for QuickPlaces and what to watch out for. The source files for our HTML Form examples are available from the IBM Redbooks Web site. See Appendix L, “Additional Web material” on page 425 for instructions on how to get the sample files.

Remember the following points when creating an HTML form manually in an HTML editor:

- Put all your code, including any JavaScript, within the <body> tag of your document. All other parts of the document, that is, the <head> and <title> tags, will be replaced by custom QuickPlace ones once the form is uploaded.
- The form tags <form> and </form> are not needed within the manually created HTML form. QuickPlace will add custom code when the form is uploaded.
- You can use QuickPlace system fields in your form. Most of them have read only values, but there are two exceptions. In our examples, we set the name of the page in a text field named h_Name, which is the system field for the page title. The other field that can be assigned a new value is the PageBody field.

6.4.1 Use QuickPlace controls in yourForm

QuickPlace provides two client-side components for users to quickly interact with the QuickPlace: the Rich text and the Upload control. Authors can format their text in the Rich text control, giving it a personal look and feel. They can easily upload documents to the QuickPlace by dragging and dropping them into the Upload control.

When using these controls, keep in mind that they only have their full functionality in Internet Explorer. Both are ActiveX controls in Internet

Explorer. Calling a Form with the Rich text control included starts a Java applet in Netscape Navigator. The Upload control displays a browse button in Netscape Navigator. The author then can select a file to upload from his computer when he or she clicks it.

Both browsers display the formatted Rich text in read mode and load the appropriate client-side component when changing into edit mode.

Internet Explorer lets the user drag a document out of the Upload control box to download or view the document, whereas Netscape Navigator displays a link the user can click to download or view the file.

The Forms provided by QuickPlace use these controls, but you can include them in your own Forms.

6.4.1.1 The QuickPlace Upload control

This example Form creates a page that includes the QuickPlace control to upload files to the QuickPlace. It also includes some fields for the user to fill in to provide information about the attached file.

The <body> tag contains the complete Form, and the <form> tags are omitted from the HTML page. We start adding fields right after the <body> tag. You can use standard HTML fields for your Form; we have used text fields, a text area and a drop-down field here.

```
<html>
<head>
<title>Upload control</title>
</head>
<body>
<table border=0>
<tr>
<td colspan=3><img src="ecblank.gif" width="300" height="1" border="0">
<tr>

       <td><b>Document Title</b></td>
       <td>&nbsp;</td>
       <td><b><input type="text" name="h_Name"></b></td>
</tr>
<tr>
       <td>Your first name</td>
       <td>&nbsp;</td>
       <td><input type="text" name="fname"</td>
</tr>
<tr>
       <td>Your last name</td>
       <td>&nbsp;</td>
       <td><input type="text" name="lname"</td>
</tr>
<tr>
       <td>Company</td>
       <td>&nbsp;</td>
       <td>
       <select name="selector">
      <option value="-- choose one --">----choose one ----</option>
       <option value="Millenia">Millenia</option>
       <option value="TheRock">TheRock</option>
       <option value="CapMan">CapMan</option>
       </select>
       </td>
</tr>
```

In the following table row we included the QuickPlace component for the Upload control. In Internet Explorer, this tag will include the ActiveX Upload control in your page, and a browse button in Netscape Navigator. In the published page, the button converts to a link to the page in Netscape Navigator, while the Internet Explorer displays the ActiveX Upload control.

```
<tr>
       <td valign="top">Give a short description<br>of your file</td>
       <td>&nbsp;</td>
       <td><textarea name="description" rows=5 cols=50></textarea></td>
</tr>
<tr>
   <td valign="top">place your file here</td>
   <td>&nbsp;</td>
   <td valign="top"><QUICKPLACECONTROL type="attachment"
name="attachment"></td>
</tr>
</table>
</body>
</html>
```

You can use the QuickPlace Upload control in any manually created HTML Form.

6.4.1.2 The QuickPlace Rich text control

This example Form creates a page that includes the QuickPlace Rich text control to add Rich text and graphic text to the page. Rich text is formatted by adding the appropriate HTML tag around it. To display the graphic text, QuickPlace uses its built-in graphics server, which is discussed in detail in Appendix 5, “The Graphics server” on page 111.

The <body> tag contains the complete form, and the <form> tags are omitted from the HTML page. We start adding fields right after the <body> tag. You can use standard HTML fields for your Form; we have used text fields and a drop-down field here.

```
<html>
<head>
<title>Upload control</title>
</head>
<body>
<table>
<tr>
<td colspan="3"><IMG src="ecblank.gif" height="5" width="200"
border="0"></td>
</tr>
<tr>
       <td><b>Document Title</b></td>
       <td>&nbsp;</td>
       <td><b><input type="text" name="h_Name"></b></td>
</tr>
<tr>
       <td>Your first name</td>
       <td>&nbsp;</td>
       <td><input type="text" name="fname"</td>
</tr>
<tr>
       <td>Your last name</td>
       <td>&nbsp;</td>
       <td><input type="text" name="lname"</td>
</tr>
<tr>
       <td>Company</td>
       <td>&nbsp;</td>
       <td>
       <select name="selector">
      <option value="-- choose one --">----choose one ----</option>
       <option value="Millenia">Millenia</option>
       <option value="TheRock">TheRock</option>
       <option value="CapMan">CapMan</option>
       </select>
       </td>
</tr>
```

In the following table row we included the QuickPlace component for the Rich text control. In Internet Explorer, this tag will include the ActiveX Upload control in your page, and a Java applet in Netscape Navigator. Both browsers display the Rich text within the relevant HTML tag. To display the graphic text, QuickPlace uses its built-in graphic server in both browsers.

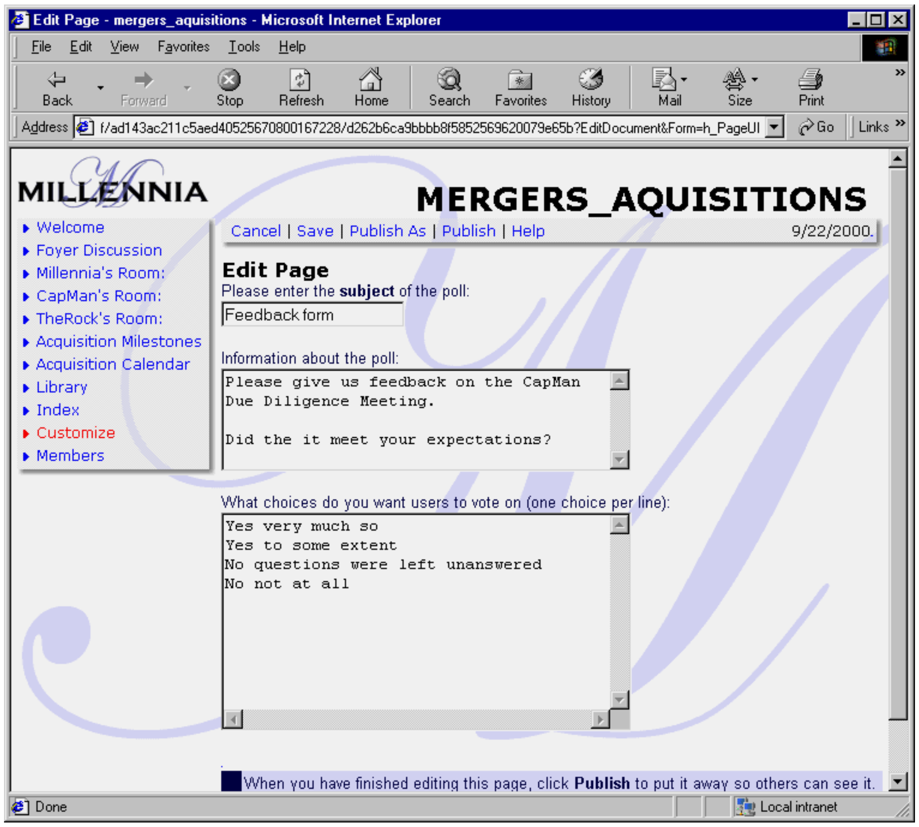
```
<tr>
   <td valign="top">&nbsp;</td>
<tr>
   <td>&nbsp;</td>
   <td>&nbsp;</td>
   <td><QUICKPLACECONTROL type="richtext" name="richtext"></td>
</tr>
</table>
</body>
</html>
```

You can use the QuickPlace Rich text control in any manually created HTML Form.

Note: At the time of writing, there was a bug with the Rich text control in read mode. When a page was being viewed in read mode, a header line with the word *Untitled* was shown over the Rich text area. If the Rich text control was placed as the first field on the HTML page, this problem did not occur.

6.4.2 Creating a poll within QuickPlace

This example lets the author create a poll when the Form is first used (as shown in Figure 67 on page 151). After clicking the submit button, any author can cast a vote in the poll. When this member views the page after voting, the results are shown, but he or she cannot cast another vote.


Edit Page - mergers_aquisitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print
Address f/ad143ac211c5aed40525670800167228/d262b6ca9bbbb8f5852569620079e65b?EditDocument&Form=h_PageUI
Go Links
MILLENNIA
MERGERS_AQUISITIONS
Welcome
Foyer Discussion
Millennia's Room:
CapMan's Room:
TheRock's Room:
Acquisition Milestones
Acquisition Calendar
Library
Index
Customize
Members
Cancel | Save | Publish As | Publish | Help
9/22/2000.
Done
Local intranet

Edit Page

Please enter the **subject** of the poll:

Feedback form

Information about the poll:

```
Please give us feedback on the CapMan
Due Diligence Meeting.

Did the it meet your expectations?
```

What choices do you want users to vote on (one choice per line):

```
Yes very much so
Yes to some extent
No questions were left unanswered
No not at all
```

When you have finished editing this page, click **Publish** to put it away so others can see it.

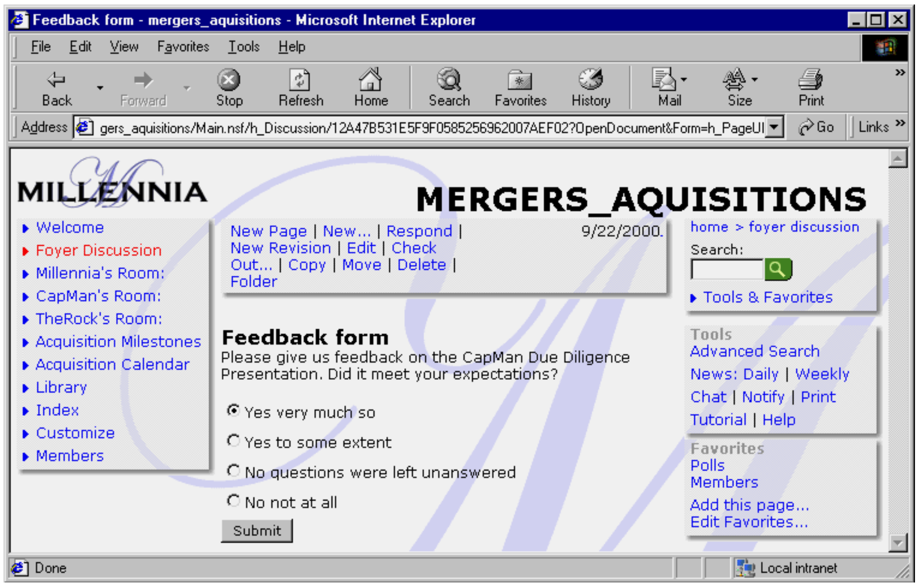
Figure 67. The Poll Form in design mode

To use this Form, the member has to be an author in the QuickPlace, as he or she actually edits the page when casting a vote.

When uploading this Form, make sure you choose the workflow type Multiple editors for it, as each vote cast edits the page the poll is based on.

This Form has three modes for displaying pages based on it.

The first is the *design* mode. In this mode, the poll is set up. The author fills in the required fields and submits the poll.


Feedback form - mergers_aquisitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print
Address gers_aquisitions/Main.nsf/h_Discussion/12A47B531E5F9F0585256962007AEF02?OpenDocument&Form=h_PageUI Go Links
MILLENNIA
MERGERS_AQUISITIONS
Welcome
Foyer Discussion
Millennia's Room:
CapMan's Room:
TheRock's Room:
Acquisition Milestones
Acquisition Calendar
Library
Index
Customize
Members
New Page | New... | Respond | New Revision | Edit | Check Out... | Copy | Move | Delete | Folder
9/22/2000.
home > foyer discussion
Search:
Tools & Favorites
Tools
Advanced Search
News: Daily | Weekly
Chat | Notify | Print
Tutorial | Help
Favorites
Polls
Members
Add this page...
Edit Favorites...
Done
Local intranet

Feedback form
Please give us feedback on the CapMan Due Diligence Presentation. Did it meet your expectations?

- [✓ Yes very much so]
- Yes to some extent
- No questions were left unanswered
- No not at all

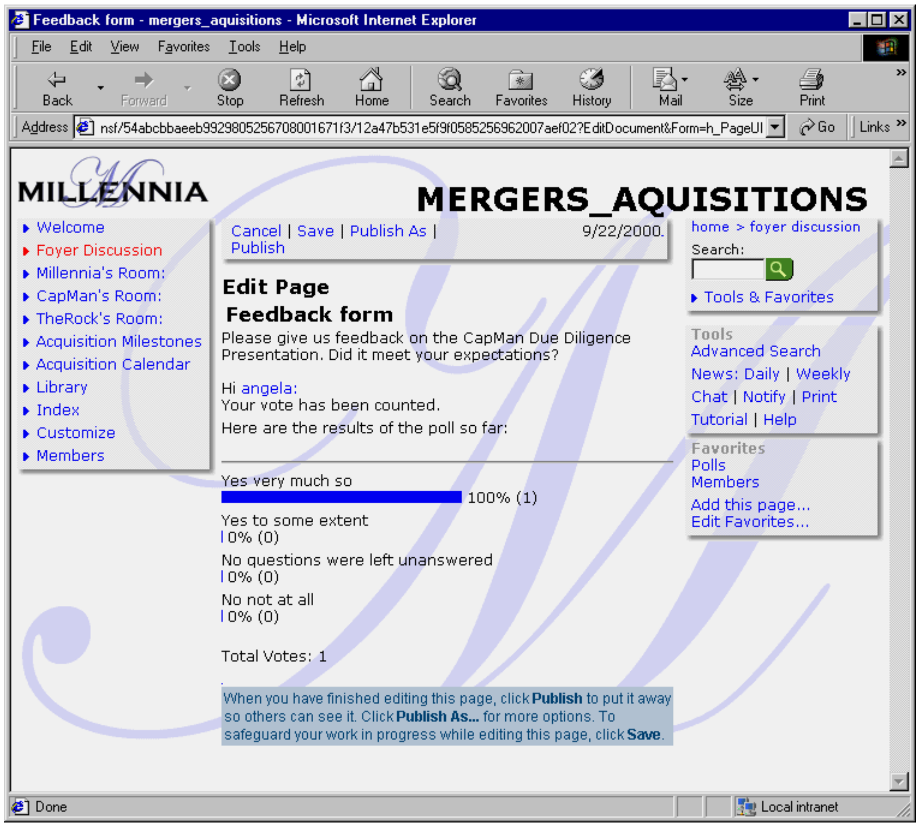
Submit

Figure 68. The poll ready to cast vote

The second mode is the actual *polling* mode. In this mode, the author casts a vote as shown in Figure 68. By doing this, he or she modifies the page the poll is based on. When casting the vote, a cookie is set on the author's machine.

The third mode displays the results of the poll so far, as shown in Figure 69 on page 153. The page is displayed in this mode when the cookie has been set on the author's machine.The cookie will expire after a preset date, which will enable the member to take part in the poll again.

Note: We use this example to show a number of techniques. It is not a production-strength poll application. A tech-savvy user can vote several times by deleting the cookie from their browser or by using another browser.



Edit Page

Feedback form

Please give us feedback on the CapMan Due Diligence Presentation. Did it meet your expectations?

Hi angela:
Your vote has been counted.
Here are the results of the poll so far:

Yes very much so
100% (1)

Yes to some extent
0% (0)

No questions were left unanswered
0% (0)

No not at all
0% (0)

Total Votes: 1

When you have finished editing this page, click **Publish** to put it away so others can see it. Click **Publish As...** for more options. To safeguard your work in progress while editing this page, click **Save**.

Figure 69. Displaying the poll results

The source files for the pollHTML Form example is available from the IBM Redbooks Web site. See Appendix L, “Additional Web material” on page 425 for instructions on how to get the sample files.

We will now walk through the coding of the poll HTML form.

We have adapted the body tag to use the StyleSheet classes we use in our Millennia Theme. Refer to 4.2, “Cascading Style Sheets” on page 58 for more information on QuickPlace Style Sheets.

```
<html>
<head>
<title>Poll</title>
</head>

<body class=h-page-bg>
```

Then, we set the hidden fields that are used within this Form:

```
<!-- Name of the Poll -->
<input type=hidden name=h_Name>

<!-- Flag to show whether the poll has been designed -->
<input type=hidden name=c_IsDesigningPoll>

<!-- The choices entered -->
<input type=hidden name=c_PollChoices>

<!-- Results of the poll -->
<input type=hidden name=c_VoteTally>

<!-- Information about the poll -->
<input type=hidden name=c_pollInformation>

<!-- Temp value of h_Authors -->
<input type=hidden name=h_Authors value="*/*/*/*">

<!-- For the poll to always use the Page skin -->
<input type=hidden name=h_SkinTypeOverride value=h_Page>
```

The fields listed in Table 26 are hidden in the poll page. We explain their usage in detail later in this chapter.

Table 26. Hidden fields on the poll page

Field name	Description
h_name	The name of the poll is set in the system field h_name. This field is set when the poll is created.
c_IsDesigningPoll	The user defined field is used to check wether the form is being designed.
c_PollChoices	This user defined field stores the poll choices
c_VoteTally	The results of the vote are stored in this user-defined field
c_PollInformation	This user-defined field stores descriptive information about the poll. This is entered on creation of the poll.
h_Authors	This system field stores information about the current author of the page.

Field name	Description
h_SkinTypeOverride	Setting the system variable h_SkinTypeOverride to h_page forces the form to be displayed using the page skin, even when it is in edit mode.

Other fields that are used within the script are shown in Table 27.

Table 27. Other fields used in Poll Form

Field name	Description
c_PollNam	The poll name is entered in this user-defined field
c_CurrentChoice	This user-defined field is displayed as a radio button, user votes by setting this field.
c_scratchPollChoices	The user-defined field the choices for the poll are originally entered into. For display, the entries of this field are checked against the user-defined field c_pollChoices

To check for the design and vote-casting mode, QuickPlace checks if the URL to the page includes the *EditDocument* command (which means the author is in Edit mode), and if the field set during design mode it is set to a value. The following code checks whether EditDocument is part of the URL:

```
function IsEditing( )
{
   return !(location.href.indexOf ('EditDocument') == -1);
}
```

If this returns true, the script checks if the value of the field *c_IsDesigningPoll* is blank.

```
function IsDesigning( )
{
   return (typeof (c_IsDesigningPoll) != "undefined" && c_IsDesigningPoll
== '');
}
```

if this returns true, the options for creating a poll are displayed.

```
if ( IsDesigning( ))
{
      // we are in design mode - let's show the options for the vote
      document.write ('Please enter the <b>subject</b> of the poll:');
      document.write ('<br><' + 'input name=c_PollName>');
      document.write ('<br><br>');
      document.write ('Information about the poll:<br>');
      document.write ('<' + 'textarea name=c_pollInformation cols=40
rows=5></textarea>');
      document.write ('<br><br>');
      document.write ('What choices do you want users to vote on (one
choice per line):<br>');
      document.write ('<' + 'textarea name=c_scratchPollChoices cols=40
rows=10 wrap=off></textarea>');
      document.write ('<br><br>');
}
```

If the check returns false, the script checks if the cookie has been set for the author. If it does not find one, the poll itself is displayed for the author to take part in:

```
else
{

   // if the current user has not voted
   // offer the vote
   if ( !HasCookie( )) {
      var HTML = "";
      HTML += '<br><span class=h-pagetitle-text>';
      HTML += h_Name;
      HTML += '</span><br>';
      HTML += c_pollInformation + '<br><br>';
      HTML += '<' + 'input type=hidden name=c_VoteTally value="';
      if (typeof ( c_VoteTally) != "undefined"){
         HTML += c_VoteTally;
      }
      HTML += '">';
      document.write( HTML);
```

It uses a loop to check for the values of the choices and creates a radio button with the field name c_CurrentChoice for each of them:

```
      var pollchoices = c_PollChoices.split(",");
      for (var i = 0; i < pollchoices.length; i++)
      {
         document.write ('<' + 'INPUT TYPE=radio NAME=c_CurrentChoice
VALUE="' + i + '">' + pollchoices[i] + "<p>");
```

The script gives you a choice to include either a submit link or button:

```
   // You have the choice of a link or a button to submit.
      //document.write('<a href="javascript:MyVoteSubmit( )">SUBMIT</a>');
      document.write('<input type="button" value="Submit"
class="h-pageSmall-text" onClick="javascript:MyVoteSubmit( )">');
}
```

In our example we used a button.

If the check found the cookie that is set after casting the vote on the author's machine, the results of the poll are displayed to the author:

```
else {

      // write out the results of the poll
      document.write ('<table width="100%" border="0"><tr><td
class=h-pagetitle-text>' + h_Name + '</td></tr></table>');
      document.write( c_pollInformation + '<br><br>');
      document.cookie.substring(countbegin , countend);
      document.write ('Hi <span class=h-toc-text>' + fnGetUser(REMOTE_USER)
+ ':</span><br>Your vote has been counted.<p>');
      document.write ('Here are the results of the poll so
far:<br><br><hr>');
```

To calculate the tally of the vote, the script creates an array and parses the values of the field *c_VoteTally* into it:

```
   var pollData = new Array( );
      var total = 0;
      for (var i = 0; i < pollOutput.length; i+=2) {
         pollData[pollData.length] = new Object( );
         pollData[pollData.length-1].text = pollOutput[i];
         pollData[pollData.length-1].count = parseInt( pollOutput[i+1]);
         total += pollData[pollData.length-1].count;
      }
```

It then creates a loop to display the value, using the image status.gif to create a graph of the votes cast so far for this option:

```
for ( i=0; i < pollData.length; i++) {
         document.write( pollData[i].text );
         document.write( "<br>");
         document.write( '<img src="$FILE/status.gif" width=' );
         if (total) {
            document.write( Math.round(pollData[i].count/total*200));
         } else {
            document.write( "0");
         }
         document.write(  " height=10> ");
```

```
            if (total) {
               document.write( Math.round(pollData[i].count/total*100) );
            } else {
               document.write( "0");
            }
            document.write( "% " + "("+ pollData[i].count + ")"+"<p>");
         }
         document.write( "<br>Total Votes: " + total + "<br>");

      }

   }
```

In vote-casting mode, the next function checks which radio button has been activated by the author:

```
function getRadioValue( radioControl)
{
   if (!isNaN (radioControl.length)) {
      var i;
      for (i = 0; i < radioControl.length; i++) {
         if (radioControl [i].checked == true) {
            return radioControl [i].value;
         }
      }
   }
   else {
      return radioControl.value;
   }
}
```

The following function then counts the vote and calculates the tally. It parses the value of the field c_voteTally into an array, increases the tally for the author's choice value, and parses the value back into the field c_VoteTally.

```
function SaveVote( myForm)
{
   var i, j;

   // unpack the vote data
   var localvoteTally = new Array();
   var pollchoices = c_VoteTally.split(",");
   for (i = 0, j = 0; i < pollchoices.length; j++, i+=2)
   {
      localvoteTally[j] = parseInt(pollchoices[i+1]);
   }

   var selectedValue = getRadioValue( myForm.c_CurrentChoice);
   if ( typeof (selectedValue) != "undefined")
      localvoteTally[selectedValue]++;

   // pack it back into the field
   myForm.c_VoteTally.value = "";
   for (i = 0, j = 0; i < pollchoices.length; i+=2, j++)
   {
      myForm.c_VoteTally.value +=  pollchoices[i] + "," +
localvoteTally[j];
      if ((i + 2) < pollchoices.length)
         myForm.c_VoteTally.value += ",";
   }
```

Then the cookie is set, preventing the author from voting again, as defined at the beginning of the script.

```
   // save the cookie
   SavePollCookie( myForm);

}
```

The remainder of the script completes the save of theForm in design mode. It checks if more that one choice was entered during design mode,

```
function SaveDesignModeParams( myForm)
{
   // read the choices

   var pollchoices = myForm.c_scratchPollChoices.value.split('\r\n');
   if ( pollchoices.length < 2)
   {
      alert ("you must enter more than one option");
      return false;
   }
```

It sets the name of the page:

```
myForm.h_Name.value = myForm.c_PollName.value;
```

It sets the poll choices:

```
myForm.c_PollChoices.value = pollchoices.join(",");
```

It resets all counters and sets c_IsDesigningPoll to a value:

```
pollchoices = myForm.c_scratchPollChoices.value.split('\r\n');
   myForm.c_VoteTally.value = "";
   for (var i = 0; i < pollchoices.length; i++)
   {
      myForm.c_VoteTally.value +=  pollchoices[i] + ",0";
      if (i < pollchoices.length-1)
         myForm.c_VoteTally.value += ",";
   }

   // we are done with design mode
   myForm.c_IsDesigningPoll.value = '0';

   return true;
}
```

The following function sets the layout of the page to the value of the system field h_StdEditPage and submits it to the QuickPlace after voting:

```
function MyVoteSubmit( )
{
   SaveVote(G_CurrentForm);
   G_CurrentForm.h_SetEditCurrentScene.value = 'h_StdPageEdit';
   G_CurrentForm.submit( );
}
```

The next function is QuickPlace-specific.You could use it, for example, to check if the user has entered a value into a specific field. In this script, we have used the function to save the form in a specific format when the author is finished designing it. We discuss QuickPlace-specific functions in 6.4.3, “QuickPlace-specific JavaScript functions” on page 161.

```
function QDK_HTMLForm_OnSubmitHandlerCallback( theForm )
{
   if ( IsDesigning( ))
   {
      return SaveDesignModeParams( theForm);
   }
}

</script>
```

This ends the JavaScript in the Poll Form. The rest of the HTML defines the URL to the image that generates the graphs to display the votes cast so far:

```
<p>
<img src="status.gif" width=0 height=0>
</body>
</html>
```

When you are finished editing the Form, save it as an HTML document and upload it to the QuickPlace. Choose **Customize**, **New form...** from the main page in the QuickPlace. Select **Imported HTML form** and click **Next**.

Name the Form and either drag and drop it into the bucket or choose **Browse** to select.

You can choose a workflow option from the list, as described earlier in this chapter.

To change the standard Publish button for your Form, select **Workflow** and choose **Simple submit**. This will add a submit button into the button bar at the top of your form when it is filled in. Click **Next** and provide a name for the submit control on the next page. Click **Next**.

Select the folder you want the pages to be published in from the drop-down list. You can add a short description of your Form, if you choose.

Click **Done** when you are finished. The form will be published to the QuickPlace.

Authors can publish pages by selecting **New...** from the button bar and selecting your Form from the list.

6.4.3 QuickPlace-specific JavaScript functions

Two QuickPlace-specific JavaScript functions to use with Forms exist in QuickPlace. They are event handlers that can be called when the Form is loaded from QuickPlace, or when a page created by the Form is submitted to QuickPlace.

QDK_HTML_FormOnLoadHandlerCallback (theForm)

This JavaScript function is called when the Form is loaded from the QuickPlace. It can, for example, set certain default field values.

The following example sets the expiration date for the page created with this Form to two months:

```
function QDK_HTMLForm_OnLoadHandlerCallback( theForm)
{
   if ( h_IsNewDoc == '1') {
      var now = new Date();
      now.setMonth( now.getMonth() + 2);
      var expirationDate = now.toLocaleString();
      expirationDate = expirationDate.substr( 0, expirationDate.indexOf( '
'));
```

```
      theForm.expires.value = expirationDate;
   }
   theForm.h_Name.focus();
}
```

QDK_HTML_FormOnSubmitHandlerCallback (theForm)
This JavaScript function is called when theForm is loaded from the QuickPlace. It can, for example, check if the user has entered data into fields that have to be filled. This example checks if a proper e-mail address has been entered by checking for the @ symbol and a dot in the string following it:

```
function QDK_HTMLForm_OnSubmitHandlerCallback( theForm)
{
   var isOk = false;
   if ( theForm.email.value.indexOf( '@') > 0 &&
theForm.email.value.indexOf( '.') > 2) {
      isOk = true;
   }
   else {
      alert( 'Please enter a valid email address');
   }

   return isOk;
}
```

These functions are very handy for performing checks or changes while loading Forms or publishing pages based on Forms. You can adapt them to perform whatever check you need for your Forms.

6.4.4 Use imported HTML page to display the News:Daily page

QuickPlace has a type of page called a *link page* where you can specify a URL for a page to open when the user clicks the link page. This is a great way to add an external page or, for example, the *News:Daily* page as the Welcome page for your Place.

Unfortunately, it does not work because the user needs to click the link page title to activate the link. However, you can still get this functionality by using an imported HTML page that redirects your browser to the desired URL.

Following is the HTML you need on your imported page:

```
<html>
<body>
<script>
```

```
location.href = "<URL of page to load>";
</script>
</body>
</html>
```

In QuickPlace 2.0, the *Whats New* pages are temporary pages and can thus only be referenced using their layout tag. References cannot be placed in the TOC (the Sidebar) or other folders. This is because creating the Whats New pages is resource-intensive and users might inadvertently place a huge load on the QuickPlace server if the page has to be opened often.

If you want to have the *News:Daily* page as the Welcome page for your Place despite the load it imposes on the server, here is how to do it:

1. Get the URL of the News:Daily page:

   ```
   In your browser, click mouse button 2 on the News:Daily link in your Place to get the
   context menu.
   ```

   c. Select **Open in New Window**. A new browser window with News:Daily should open.

   d. In the new browser window, copy the part of the URL after the host name to the clipboard.

      The copied part should look similar to this (include */QuickPlace/Placename/...*):

      ```
      /QuickPlace/Blue_Lagoon/Main.nsf/h_Index/......,h_SetWhatsNewDays;1
      ```

2. Paste the URL link into the HTML link page shown above. Save the HTML file. We called ours *linknew.htm*.
3. In your Place, create a **New -> Imported Page**. Click **Next**.
4. Drop or pick *linknew.htm* to the upload control, give the page a title, and click **Publish**.
5. Select to have the page place at the top of the TOC (Sidebar) and click **Next**.

That's it; users of your Place will now see the Daily Whats New page as soon as they enter the Place. But keep in mind--this is an expensive feature in terms of server resources.

Summary

In this chapter, we have explained how to create your own Forms in QuickPlace. We described three ways to define Forms, by creating them using a field provided by QuickPlace, or by uploading forms previously created in a MS Office product or in an HTML editor.

We also explained how to adopt workflow in a QuickPlace Form.

Then, we analyzed a JavaScript creating a Poll within QuickPlace as an example on how to use JavaScript in a Form.

Finally, we discussed the QuickPlace Rich text and Upload controls and the QuickPlace-specific JavaScript functions.

Chapter 7. PlaceBots

In this chapter we discuss how to add application logic and automation to Places using PlaceBots. First we will define a PlaceBot and explain how a PlaceBot relates to a Domino agent, together with a giving brief overview of how to write one in either Java or LotusScript. Then we will look at the following examples:

- An automatic sitemap PlaceBot (called *Mapperizer*)
- A Placebot that moves documents to different folders based on words in their topics (called *Mail Room Attendant*)
- A server status application where the main part is a scheduled Domino agent (called *TheXRay*)

7.1 What is a PlaceBot

A PlaceBot is a Domino agent, written either in Java or LotusScript, that automates a task. Using a PlaceBot, you can access, process, and manage the data in a Place. For example, you could use a PlaceBot to send e-mail to members of your Place that notifies them when a document of interest is edited. You can create a PlaceBot that runs when a Form is submitted, or on a scheduled basis. You can also run a PlaceBot manually.

You create PlaceBots using LotusScript or Java to manipulate the Domino back-end object classes. For complete documentation on the Domino Object Model and how to work with objects using LotusScript or Java, see the *Lotus Domino R5 Programming Guide*, available as a downloadable file from:

`http://notes.net/notesua.nsf`

For more information on how the elements in a Place map to objects in the Domino Object Model, see the QuickPlace Developer's Toolkit, available as a downloadable file from:

`http://www.quickplace.com/devzone`

It is possible to write, debug, and compile Java code for a PlaceBot in a Java development tool, such as VisualAge for Java or Symantec Visual Cafe. You can then import the .java file, or compile and import a .class or .jar file. It is also possible to write Java or LotusScript code in any editor and import the resulting files into your QuickPlace. When you upload Java Placebots, they are automatically compiled, and a syntax check is done on them.

We recommend having PlaceBots created in the Domino Designer Integrated Development Environment (IDE), exported as LSS or Java files, and uploaded into a QuickPlace. This is not only faster to iterate through changes, but it also avoids crashing your HTTP server. If you write bad commands when you are debugging in the Domino Designer, you have the ability simply to use control break keys.

7.1.1 Triggering PlaceBots

QuickPlace has a page in the Customize area where you can specify settings to control how and when the PlaceBot runs. These settings are similar to the Domino agent settings, but not identical.

There are two ways of triggering PlaceBots: scheduled PlaceBots, which are triggered by a timer event, or Form PlaceBots, which are triggered by the creation of documents, based on the Form type.

7.1.1.1 Scheduled PlaceBots

Scheduled PlaceBots are processed by a server task called the *agent manager*, and have the access rights of the QuickPlace Manager. For scheduled PlaceBots, the options are:

1. Monthly, Weekly, Daily or Minute Intervals
2. Act on all docs or new/modified docs
3. Optionally act on docs in a specific folder
4. Optionally specify start/end dates, enable/disable

7.1.1.2 Form Placebots

Form Placebots are triggered by the creation of a document based on any of the user Forms in a QuickPlace. For example, if we create a new HTML Form, you can specify that the PlaceBot is to run on documents created with that Frm. If PlaceBots are triggered by document creation, they have the same access rights as the person who created the document.

7.1.1.3 The difference between PlaceBots and Domino agents

PlaceBots are in effect, Domino agents. The difference between an agent and a PlaceBot is that a PlaceBot is installed into a QuickPlace via the QuickPlace upload control. An agent is created (and “installed”) using Domino Designer.

There is one other subtle difference, and that is that a PlaceBot can be created in any text editor, then uploaded. In theory you do not need to have Domino Designer. However, the reality is that without the Domino Designer client, it is very slow to test and debug your code.

The advantage of PlaceBots over agents is that they can be stored as small LSS, Java archive (JAR), or Java Class files on your hard disk, and installed quickly on any QuickPlace, even when you do not have administrative access to the server. You just need Manager access to the Place where you want to install the PlaceBot.

7.1.2 PlaceBot security

If PlaceBots are run on a scheduled basis, they inherit the access control of the server. When Placebots are run manually, they have the access rights of the person triggering them. In other words, you must have Manager access to create, edit, copy, delete, or run PlaceBots manually.

If a badly coded PlaceBot is installed in a Place, this could cause the entire QuickPlace server to fail.

If a PlaceBot is badly coded, is extremely processor-intensive, or is set to run very often, this can slow down the entire QuickPlace server.

Because of security and performance considerations, some administrators might choose to disable the PlaceBot feature set entirely. Consult your server administrator about PlaceBot policy in your organization.

7.1.3 How to write a PlaceBot: Quick Start

The following section is for experienced programmers who want to get stated immediately writing PlaceBots, without having to read any instruction manuals.

To write these PlaceBots, we first of all created a dummy Place using the Domino Designer. To do this we took the following steps:

1. Create a Place (we called ours BotTesterizer).
2. Create as many examples of documents, folders, links, and so on that your PlaceBot will be referencing. In this chapter we describe the Mapperizer PlaceBot. This PlaceBot maps all documents contained in a room.

   When we created the Mapperizer PlaceBot, we created a Response folder, a normal folder, and then 15 documents that appear in many different views within the Place. We chose to add many examples of pages, folders and so on because they were necessary to seeing if the PlaceBot would find all of the necessary elements in a Place. We suggest making a Place that contains one example of all the elements a database can contain that is relevant to your PlaceBot.
3. Close down the QuickPlace server.

4. Create a copy of the NSF files that were created by QuickPlace.

   In many cases, you may only need to create one NSF file. However, when you are creating a PlaceBot that tests functionality between Rooms, you will need to create multiple NSF databases. Be sure to use a *copy* of the database, not a *replica*. If you use a replica, you can experience problems. For example, your local interim changes may be transferred to a production database. Making a copy makes life much easier.

   In most cases you will only need to use the main.nsf. You will find this file in a data directory with the same name as your Place. We found ours in:

   ```
   C:\Lotus\Domino\Data\QuicPlace\BotTesterizer\main.nsf
   ```

5. Open the NSF file in Domino Designer and start writing a Domino agent.
6. Test the PlaceBot in the NSF, until you have a working Domino agent.

   This will not simulate all of the conditions of a real Place, but it will simulate most of them. You may still need to debug the PlaceBot after it is uploaded, but it allows you to resolve most of the small bugs.

7. Export the agent and save java agents as a .Java , .JAR , .Class or .ZIP file. If you are writing a LotusScript file, save it in .LSS format.

   To export a Java file, press the Export button at the bottom of the Designer pane. To export a LotusScript file, right-click the Designer pane and select Export. In the options that appear, if you are not sure, choose All Sections.

8. Make sure the QuickPlace server is started and return to your original Place.
9. Go into **Customize -> PlaceBots -> New PlaceBot** to create and test your PlaceBot. The steps for creating a PlaceBots are briefly:
   - Specify when the PlaceBot should run.
   - Import the files for the PlaceBot.
     - For a LotusScript agent, import a single .lss LotusScript source file.
     - For a Java agent, import one or more .java, .class or .jar files.
   - Click **Done** when you have filled in all the required fields.
10. For small changes, we found it easier to open the LSS file in a text editor, then copy the changes back into the original LotusScript agent. For more complex changes, we went back into the Domino Designer.

7.2 Java PlaceBots

QuickPlace 2.0 supports Java, Version 1.1.8 and later. A PlaceBot written in Java may consist of one or multiple files. A Java PlaceBot file must contain a class that extends the Domino Java agent class AgentBase.

Java PlaceBot files can be of the following types:

- .java files contain Java source code.

  These files are compiled on the QuickPlace server when the PlaceBot is submitted through the browser.

- .class files are Java object files produced by compiling the .java files.

  Since these are already compiled, the files do not need to be compiled when they are submitted to the Quickplace server. To compile your .java source agent files into .class files locally on your machine, you will need a copy of the Notes.jar files locally. Notes.jar is included with each QuickPlace server installation. If you do not have access to this file, ask your QuickPlace server administrator to make it available to you.

- A .jar file is a zipped, or compressed, collection of files.

  The .jar file generally contains one or more .class files and any other files (for example, graphic files) the PlaceBot requires.

A PlaceBot extends the AgentBase class, which extends the NotesThread class. The class that contains the PlaceBot code must be public. The entry point to the functional code must be public void NotesMain().

For more information on writing code for a Java PlaceBot, refer to the information on writing Java agents in the Domino Designer Programming Guides, available as downloadable files from:

`http://www.notes.net/notesua.nsf`

7.2.1 Java PlaceBot example

This Java agent writes the name of each document it processes to the log.

```
import lotus.domino.*;
import java.util.*;

public class LogTitles extends AgentBase {

   public void NotesMain(){

      try {

         Session s = this.getSession();
         AgentContext ctx = s.getAgentContext();
         Database db = ctx.getCurrentDatabase();
```

```
            // Prepare the agent log
            Log log = s.createLog("Log");
            log.openAgentLog();
            log.setLogActions(true);

            // Get all the unprocessed documents
            DocumentCollection dc = ctx.getUnprocessedDocuments();
            if (dc.getCount() == 0) {
                return;
            }

            // Loop thru the documents and print the title of each document.
            Document doc = dc.getFirstDocument();
            log.logAction("Count of documents = " + dc.getCount());
            while (doc != null) {
                log.logAction(doc.getItemValueString("h_Name"));
                doc = dc.getNextDocument();

                // Mark all the documents as processed.
                dc.updateAll();
            }
        }

        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

There are several development tools for writing Java PlaceBots. On the lowest level, you can write PlaceBots in a text editor and use QuickPlace as your test environment. We have found that this method has a minimal client requirement, but has a very long development cycle, and can cause the server to crash if your code is “buggy”.

It is also possible to use third-party Java development tools, such as Visual Age for Java. When you use this method, it is necessary to include the Notes.jar file in the CLASSPATH.

We have found using the Domino Designer to be a good alternative. It provides convenient access help tools and allows you to quickly test your changes.

7.3 LotusScript PlaceBots

Creating PlaceBots using LotusScript is essentially the same as writing an agent for a Domino application. First you need to know your programming language, and then you need to understand the application you are extending.

It is assumed that you have some LotusScript knowledge, and in this section we focus on how to extend the QuickPlace application. We describe how to start writing LotusScript PlaceBots, leveraging existing knowledge, and re-using existing code.

There are two alternatives for writing LotusScript (LSS) Placebots. It is possible to write LSS PlaceBots in a text editor and test the PlaceBots in the QuickPlace. We have found this to be a good approach for *finetuning* PlaceBots.

However, we recommend instead that you use the Domino Designer, test the PlaceBot in a Domino NSF database, and upload the LSS file. This method has several advantages. The major advantage is speed of development, because in the Domino Designer development environment, you have syntax checking, debugging, and the ability to access LotusScript Class help files very quickly.

The following section uses examples of LotusScript that show the process of creating a PlaceBot and expose the Object Model.

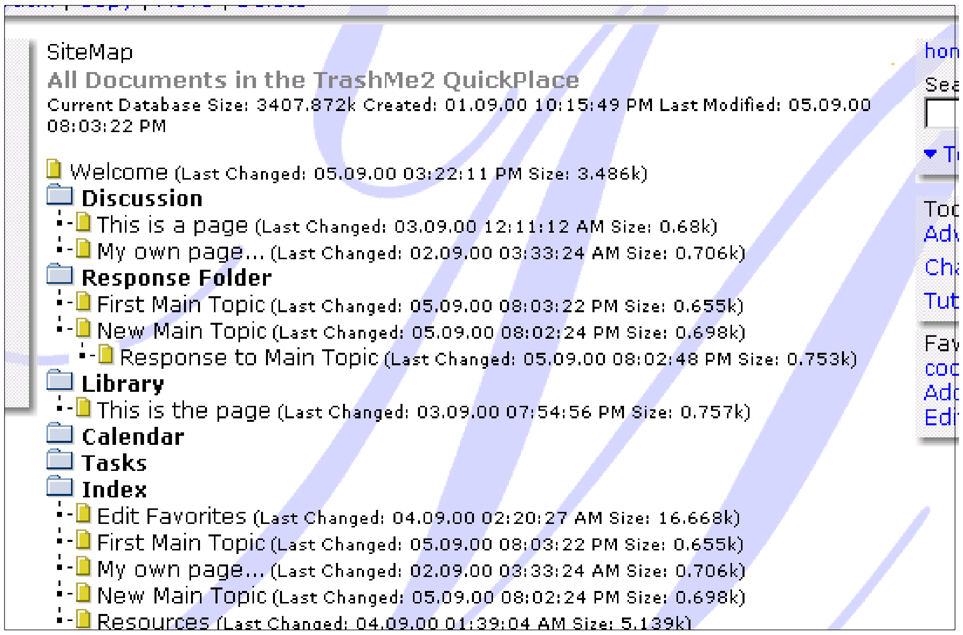
7.4 A site map PlaceBot

Our client, Millennia, wanted to be able to see all documents in the current room at once. They were also interested in seeing statistical data about the room, while still maintaining their corporate look and feel.

The option for mapping links to rooms was discussed but decided against, because it would allow members without access to a particular room to see a link to that area. It must be stressed that the *list of items in the room presented by the PlaceBot are only links, and the user will still need access rights to read the document*. However, QuickPlace's established structure is to not show links to which the user does not have access. The other reason for not putting room links in was that the map was essentially for the purpose of showing documents.

Links were kept in, as we decided that this PlaceBot would make a good basis later for a special HotLinks page, and we wanted to keep the code.

Millennia was also interested in being able to impose QuickPlace Theme-like technology on the page. They wanted to be able to change the images, and the look of the text, without having to involve a programmer.



SiteMap

All Documents in the TrashMe2 QuickPlace

Current Database Size: 3407.872k Created: 01.09.00 10:15:49 PM Last Modified: 05.09.00 08:03:22 PM

Welcome (Last Changed: 05.09.00 03:22:11 PM Size: 3.486k)
Discussion
- This is a page (Last Changed: 03.09.00 12:11:12 AM Size: 0.68k)
- My own page... (Last Changed: 02.09.00 03:33:24 AM Size: 0.706k)
Response Folder
- First Main Topic (Last Changed: 05.09.00 08:03:22 PM Size: 0.655k)
- New Main Topic (Last Changed: 05.09.00 08:02:24 PM Size: 0.698k)
  - Response to Main Topic (Last Changed: 05.09.00 08:02:48 PM Size: 0.753k)
Library
- This is the page (Last Changed: 03.09.00 07:54:56 PM Size: 0.757k)
Calendar
Tasks
Index
- Edit Favorites (Last Changed: 04.09.00 02:20:27 AM Size: 16.668k)
- First Main Topic (Last Changed: 05.09.00 08:03:22 PM Size: 0.655k)
- My own page... (Last Changed: 02.09.00 03:33:24 AM Size: 0.706k)
- New Main Topic (Last Changed: 05.09.00 08:02:24 PM Size: 0.698k)
- Resources (Last Changed: 04.09.00 01:39:04 AM Size: 5.139k)

Figure 70. The SiteMap Page as it appears in the Millennia QuickPlace

7.4.1 Functional features for the Mapperizer PlaceBot

The PlaceBot has the following list of functional features:

- Graphical display of folder/page relationships
- Alphabetical list of pages
- Theme-like customization of the map page
  - Easily modification of folder and page images
  - Full control over fonts, sizes, colors and so on
  - Ability to wrap the document in any HTML required
- Selective page mapping - Pages and Links, but not Room links
- Map document could be switched off if decided by the Manager
- Map document could appear wherever the Manager decided
- Map document would be updated whenever a new Page document was added

7.4.2 Components in the Mapperizer

Together with the actual PlaceBot code our solution has the components listed in Table 28:

Table 28. Components of the Mapperizer

Component	Description
Mapperizer.lss	The placebot Code. This is the actual PlaceBot that will perform the functions. To change this PlaceBot, you can use a text editor. For more complex changes, it is best to copy/paste it into a Domino agent in the Domino Designer.
SiteMap.htm	This file positions and presents the data. This file works in the same way a layout file used in Themes works. For a better understanding of Themes, see Chapter 4, "Creating Themes" on page 47. SiteMap.htm can be modified or replaced with a customized file. We find it easier to simply modify this file for each new application. This file must be named SiteMap. If you want to change this name, you need to change the variable sMapDocName in the Mapperizer PlaceBot. This string can contain spaces.
Folder.gif , Page.gif , SubPage.gif , Indent.gif	The images that appear in the SiteMap page. To change these, simply modify the images in an image editor program and save them with the same name. These images cannot be renamed unless you modify the Mapperizer PlaceBot.
Mapperizer.css	Optional Style Sheet definitions for the Mapperizer PlaceBot, containing special tags for the SiteMap page. When creating your own Theme, add these to your CSS file, normally called StyleSheet.css.

7.4.3 Installing the Mapperizer Placebot and files

The source files for our Mapperizer solution are available for download from the IBM Redbooks Web site. Before looking at the code in the PlaceBot, we will describe the steps to install PlaceBot so you can look through the actual code yourself while reading through this chapter. See Appendix L, "Additional Web material" on page 425 for information on how to get the sample files.

In the following section we describe:

- Importing the sitemap HTML layout page
- Uploading the LotusScript PlaceBot
- Testing the PlaceBot

7.4.3.1 Create the SiteMap QuickPlace page

1. Click the **New... button** to create a new page.
2. Select **ImportedPage**.
3. Name the Page "SiteMap". You must call it this or the Mapperizer will not work.
4. Turn off: Show the title, author and date on page?.
5. Press the Browse button to open a file dialog box.
6. Find and open the SiteMap.htm file.
7. Click **Publish**.
8. Choose a location for the file. We recommend placing it in the sidebar near the Index, as the last elements here are used for managing Places, las the Mapperizer is.
9. Click **Next** to save the document
10. Look at the document. If this is the first time you have installed the Mapperizer, the following message will appear in the page:

    ```
    Run the Mapperizer Placebot to see a site map here...
    ```

11. If you cannot see the page, make sure that you have followed all of the previous steps.

7.4.3.2 Upload the Mapperizer.lss Placebot

1. Enter the Customize section of your Place.
2. Scroll down and click the PlaceBots link, and you will be taken to the PlaceBot creation area.
3. Call the PlaceBot Mapperizer.
4. Enter a description. We entered: `Creates a map of a QuickPlace room.`
5. Enter the section entitled: When should this PlaceBot run?. Go to the section entitled: When a form is submitted:, and choose **Page**. Mapperizer can also be run on schedule, but this will make the content of the PageMap page not as accurate.
6. Go to section 4 and click the button that looks like a folder called **Select File(s)**.
7. Locate Mapperizer.lss PlaceBot, and click **Open**.
8. Click **Done** and the PlaceBot will be uploaded.
9. You will be taken into the PlaceBots Area. Make sure that you can see the Mapperizer agent listed on the page with a radio button next to it.

7.4.3.3 Test the PlaceBot

1. If you are not in the PlaceBots area of your Place, enter the Customize section of your Place. Scroll down and click the PlaceBots link; you will be taken to the PlaceBot creation area.
2. Click the radio button next to the Mapperizer link.
3. Click **Run Placebot**.
4. A message will appear telling you that the PlaceBot was set to run on schedule, and that it may cause problems if you run it manually.
5. The Log page will appear, telling you of the results of attempting to run the agent. If it was successful, you will see the following message:

```
Started running agent 'Mapperizer' on 09/20/2000 01:02:15 AM
09/20/2000 01:02:17 AM: Mapperizer: IBM Redbook Example
09/20/2000 01:02:17 AM: Mapperizer: Run agent: Mapperizer
09/20/2000 01:02:24 AM: Agent Mapped 18 documents
Ran LotusScript code
Done running agent 'Mapperizer' on 09/20/2000 01:02:25 AM
```

6. If you have not loaded the SiteMap page, the following error message will appear:

```
Started running agent 'Mapperizer' on 09/20/2000 12:33:49 AM
09/20/2000 12:33:49 AM: Mapperizer: IBM Redbook Example
09/20/2000 12:33:49 AM: Mapperizer: Run agent: Mapperizer
09/20/2000 12:33:49 AM: Mapperizer: Document SiteMap not found
Ran LotusScript code
Done running agent 'Mapperizer' on 09/20/2000 12:33:49 AM
```

7.4.4 Description of the code in the Mapperizer PlaceBot

Why document this PlaceBot? The significance of this PlaceBot is that it gives a good example of how you can access the QuickPlace Object Model. It is also a good starting point for writing your own PlaceBots. The lengthy documentation should help you to be able to create PlaceBots that differ from this one by being able to understand the working environment.

The Mapperizer PlaceBot starts off by dimming (declaring the variables) related to the session, the database, and its major structures such as Views and key documents. We divided our dims into two groups: dims for Domino-related objects, and writing dims, which are used in content creation, mostly via strings. The lines here where you see an ellipsis (...) indicate where we have removed text to make this section as concise as possible:

```
'notes dims
Dim ses As New NotesSession
Dim ndb As NotesDatabase
Dim docInTOC As NotesDocument
Dim viewTOC As NotesView

'writing dims
Dim sSysName As String
```

```
...
Dim iIndentDistance As Integer
```

The PlaceBot Customize page allows you to read the log created for each PlaceBot. The following argument sets up logging. Then it prepares the normal error handling that we always put into all our applications:

```
Dim nLinkCount As Integer
Dim logAgent As New NotesLog( ses.CurrentAgent.name )
logAgent.LogActions = True
Call logAgent.OpenAgentLog
nLinkCount = 0
Call logAgent.LogAction("Mapperizer: Run agent: " &
    ses.CurrentAgent.name )
On Error Goto lblLogError
```

This section sets the document-related variables. These sets are related to finding documents, working from the database, down to the individual SiteMap document.

If the SiteMap page is not found in the Place, then the PlaceBot quits. This is to allow a bit of flexibility in the order in which you install the PlaceBot; you do not get errors if you do it out of order, delete the site map page, or name the site map page wrongly.

We decided not to have the PlaceBot refresh a Page instead of replacing it, because creating a document which was always used allowed us to store image components in a document, and have them referenced each time.

This also allowed us to use a layout file for the Site Map page. The layout file is the SiteMap.htm. This HTML wraps the map and can be modified, Place by Place, without having to change the PlaceBot.

In its current form, the SiteMap PlaceBot and Page separate LotusScript and HTML scripting, thus making the overall programming of the PlaceBot simpler.

```
sMapDocName = "SiteMap"
Set ndb = ses.CurrentDatabase
Set viewCurrent = ndb.getView( "h_Index" )
Set docReport = viewCurrent.GetDocumentByKey( sMapDocName )
If docReport Is Nothing Then
    Call logAgent.LogAction("Mapperizer: Document " & sMapDocName & " not found" )
    Exit Sub
End If
```

Next we need to find the Table of Contents, and set this as a special view. In the looping functions that follow, it gets each element in the viewTOC, and then performs a number of calculations on that TOC element. Some TOC elements may be views, in which case that “sub view” is stored in the variable named viewCurrent.

```
Set viewTOC = ndb.getView("h_Toc")
```

To start creating the URL strings for each entry, the following dims and sets are made. The sDbPath now is a string formatted for use within a URL; in other words, its backslashes are replaced by forward slashes.

ASCII character 47 is a forward slash, and character 92 is a backslash.

```
Dim sFSlash As String
Dim sBSlash As String
Dim sDbLabel As String
Dim iSlashPos As Integer
sDbPath =  ndb.FilePath & "/"
sFSlash$ = Chr(47)
sBSlash$ = Chr(92)
iSlashPos = Instr(1, sDbPath$, sBSlash$)
While iSlashPos <> 0
    Mid$(sDbPath$, iSlashPos, 1) = sFSlash$
    iSlashPos = Instr(1, sDbPath$, sBSlash$)
Wend
```

To introduce images onto the current page, we need to create a URL that references the current QuickPlace document, then the images that are attached to it. These images are automatically uploaded when the SiteMap.htm file is uploaded (see 7.4.3.1, “Create the SiteMap QuickPlace page” on page 174). The URL below uses the *h_Index* view in the current Place as an index. The key used to locate the correct document in the h_Index view is supplied via the variable *sMapDocName*.

Note: h_Index is the alias name of a view that allows you to look up all documents in the Place using their ID. For example, to get an image:

```
<img
src=”http://quickP.com/quickplace/MyPlace/Main.nsf/h_Index/PageMap/$FILE/Page.gif?OpenEl
ement”>
```

To simplify this, we’ve kept the same path structure but made it relative:

```
<img src=”../../PageMap/$FILE/Page.gif?OpenElement”>
```

In the following script, you may notice that the name of the PageMap document is replaced by a variable. This will make it easier to change the name of the file.

The names of the GIF images names are hardcoded. If the developer wants to modify the SiteMap look, it is easier to just create a new image and save it with the same name.

```
'writing sets
sFolderImg = |<img src=../../h_index/| & sMapDocName & |/$FILE/Folder.gif?OpenELement
border=0
    width=20 height=13>|
sPageImg = |<img src=../../h_index/| & sMapDocName & |/$FILE/Page.gif?OpenELement
border=0
   width=13 height=13>|
sPageSubImg = |<img src=../../h_index/| & sMapDocName & |/$FILE/PageSub.gif?OpenELement
border=0
   width=29 height=13>|
sIndentImg = |<img src=../../h_index/| & sMapDocName & |/$FILE/Indent.gif?OpenELement
border=0
   width=13 height=13>|
sHeadingStyleTag = "<span class=h-mapHeading-text>"
sStyleTag = "<span class=h-map-text>"
sFolderStyleTag = "<span class=h-mapFolder-text>"
sDetailStyleTag = "<span class=h-mapDetail-text>"
sEndStyleTag = "</span>"
sIndentConcat = ""
```

The next few lines set up more HTML, the first of which is a header line. Note that the HTML <HEAD> tag is written over when a HTML file is imported, and so the sHTMLHead variable's contents are not used unless you are writing to the disk as an HTML file (see end of PlaceBot text). This is included for debugging purposes only.

The sHTML01 tag is added to the document before the map HTML. Note that this is not at the start of the entire document, but at the start of the Map. To put code into the true start of an HTML document in a Place, you need to manipulate the Domino objects directly, using the Domino Designer.

If you want to insert JavaScript or Style tags in your site map document, it is recommended that you put them in sHTML01 string variable.

The sHTML02 tag is added at the end of the map HTML. Note that this HTML is not at the absolute end of the HTML stream for the page; it appears at the end of the map section.

We like using the pipe character to define strings because it allows you to put in un-escaped quote characters, but more importantly the strings appear as you type them when you look in the source code of your browser, making it easier to debug.

```
sHTMLHead = |<html><head><link rel=stylesheet type="text/css"
   href="Mapperizer.css"></head>| 'only for debug
sHTML01 = |<!-- start -->| 'put JavaScripts here or the imported document
sHTML02 = |<!-- end -->| 'this will be concantenated at the end
```

Now we can start looping through the Table of Contents. The view h_Toc is very important to this PlaceBot, as it is the main guide to locating relevant folders and documents.

It is also possible to use the h_Index to find all documents, and look at their properties. However, this becomes cumbersome because you need to check for too many values, in order to sort them correctly. For this reason, in this situation we recommend that you use the h_Toc, because it is much smaller and the items appear in the correct order. If you want to get all documents, the h_Index is a good view to use.

We also see here that if a document cannot be found in the TOC view, the PlaceBot exits. This is because if we are not able to find the TOC, errors will occur if we continue.

```
sMainText = sHTML01
Set docInTOC = viewTOC.GetFirstDocument
If docInTOC Is Nothing Then  'make sure the TOC is found
    Call logAgent.LogAction("Mapperizer: TOC Not Found" )
    Exit Sub
End If
```

In this section we see the visible text beginning to be created, first the title of the page, including the Place name.

Note: The StyleTag is a CSS span instruction to help with formatting.

```
sMainText = sMainText & sHeadingStyleTag &
        "All Documents in the "
sMainText = sMainText & ndb.Title & " QuickPlace" &
        sEndStyleTag
```

Now a few of the database statistics are added to the main text string. If you want to add more information about the Place, it is recommended that you put them here...

```
sMainText = sMainText & "<br>" & sDetailStyleTag & "Current Database Size: "
        & Cstr(ndb.Size / 1024) & "k "
sMainText = sMainText & sDetailStyleTag & "Created: " &
        Cstr(ndb.Created)
sMainText = sMainText & " Last Modified: " & Cstr( ndb.LastModified ) &
        sEndStyleTag & "<br>"
```

Then start cycling through the documents in the view named TOC. The TOC is a very special view in a QuickPlace and helps us to understand a lot about QuickPlace generally.

The TOC view contains a list of links that refer to items the users of the Place have chosen to display there. For example, in our Place we have a Welcome document, a Discussion view, a link, a slide show and some other QuickPlace tools such as Customize.

Each of these elements is represented in the TOC by a data note. The only thing that the objects here have in common is that they have the field h_IsInTOC containing a value of "1" (text), and that it is properly published (is not in draft mode and does not contain any $Conflict fields or documents that are in draft mode).

Published items in QuickPlace are distinguished by an important field: the h_Type field. If this field contains a “0” it is a document; if it contains a “1”, it is a view, and so on. In the example below, we find the value of “0” in this field and thus begin dealing with this item as a document.

```
While Not ( docInTOC Is Nothing )
If docInTOC.h_Type(0) = "0" Then 'it is a doc
    Set docTemp = docInTOC
```

Now that we know that we have found a document, we can start building the text string for that element’s listing.

It would be a little strange if the site map page was visible in the map, so we make sure that it is skipped in the looping process. Remember that the sMapDocName variable has the document’s name assigned to it.

```
If docTemp.h_Name(0) <> sMapDocName Then
```

Another type of document that is handled a little differently is the Link type document. QuickPlace knows a document is a Link if the field h_URLPointer is not empty. So if the h_URLPointer field has content, we build the URL string using that field’s content.

```
If docTemp.h_URLpointer(0) <> "" Then 'it is a link
    sMainText = sMainText & "<br>" & "<a href=" & docTemp.h_URLpointer(0) & ">"
```

Otherwise, build the URL using the main view of any Main.nsf database, the h_Index view. The h_Index view is the most important view for creating unique resource locators (URLs). There is more information on this in Chapter 8, “QuickPlace Object Model” on page 217.

The following line uses the Universal Identifier of the page being referenced. This value can be found in the first, visible, sorted column in the h_Index view. If you do not understand how Domino URLs are built, you can look in the Domino Designer help database listed under “Domino URLs”, or look at the overview in Appendix D, “Domino URL commands” on page 371.

```
Else
    sMainText = sMainText & "<br>" & "<a href=../../h_Index/" & Cstr(docTemp.UniversalID)
    & "?OpenDocument>"
End If
```

Here the script continues building the text string for that element’s listing.

```
sMainText = sMainText & sPageImg & sStyleTag & docTemp.h_Name(0) & sEndStyleTag
sMainText = sMainText & sDetailStyleTag
sMainText = sMainText & " (Last Changed: " & Cstr(docTemp.LastModified)
sMainText = sMainText & " Size: "
If docTemp.Size < 1024 Then
    sMainText = sMainText & "0"
End If
sMainText = sMainText & Cstr( docTemp.Size / 1024 ) & "k"
sMainText = sMainText & ")" & sEndStyleTag & "</a>"
nLinkCount = nLinkCount + 1
         'end standard formatting
End If
```

The next *Elseif* operator executes if a view is found. It must be remembered that we are not actually looking at a list of views, but a list of data "notes". Each of these data notes creates a link to a database element; in this case a view. QuickPlace's definition of a View note is that the h_Type field is 1.

We first of all need to find the internal name of the view being referred to. This is done by finding the value of the h_SystemName field. We then store that in the sSysName variable. The sSysName now contains the internal name of the view. This variable helps us to find the current folder or view.

```
Elseif docInTOC.h_Type(0) = "1" Then 'it is a folder/view
   sSysName = docInTOC.h_SystemName(0) 'get the internal name of the view
   'if it is a link to a QuickPlace tool page
```

Then we test the sSysName again to see if we are looking at the Tailor section (called the "Customize" section in the user interface, or the Members area. Because we are only trying to map documents, these two are ignored. We have left this test in here as it is in case you want to revise this agent and map them as well.

```
   If ( sSysName = "h_Tailor" ) Or ( sSysName = "h_Members" ) Then
      'do not do anything
      'add code here if you want to map these docs as well
```

We now know that we are looking at a folder, and it is one of the folder types that we want to map. We will now check whether the folder is of the type response folder to know if some form of indentation is suitable. The iNested variable is set to 1 if it is a response folder.

```
Else
   iNested = 0
   If docInTOC.h_FolderStyle(0) = "5" Then 'it is a response folder
      iNested = 1
   End If
```

We now put the view name into the viewCurrent variable. Note the URL to the view used here. We do not go to database/view, but rather database/h_Index/viewDesignNoteID. This another example of how the QuickPlace Object Model works with Design Notes. Referencing the design note will redirect you correctly to the view. This section of the script does exactly that, and writes some more text to the sMainText variable, which will later be printed out as the map page content.

```
Set viewCurrent = ndb.getView( sSysName )
sMainText = sMainText & "<br><a href=../../h_Index/" & Cstr(docInTOC.UniversalID)
   & "?OpenDocument>"
sMainText = sMainText & Chr(13) & Chr(9) & sFolderImg
sMainText = sMainText & sFolderStyleTag & docInTOC.h_Name(0) & sEndStyleTag & "</a>"
```

The folder has been found, and written to the sMainText variable. Now it is time to get all the documents in the view. To do this, the docTemp variable is used to gather all documents in the view.

The code in this section is similar to the above, when we find documents in the top level of the TOC. However, there are several subtle differences. The main one that the user sees if that they are indented to one extent or another. Normal documents are indented one space (13 pixels), but pages in response folders that are responses are indented a further 13 pixels.

```
Set docTemp = viewCurrent.GetFirstDocument
While Not (docTemp Is Nothing)
    If docTemp.h_Name(0) <> sMapDocName Then
        sMainText = sMainText & "<br>"
        If docTemp.IsResponse And iNested Then
            iIndentDistance = 13
            sIndentConcat = "<img src=blank.gif width=" & Cstr(iIndentDistance)
                & " height=1 border=0>"
            sPrevSetParentUnid = sThisSetParentUnid
        Else
            iIndentDistance = 0
            sIndentConcat = ""
        End If
        sMainText = sMainText & sIndentConcat
```

Once again we see the handling of link documents but now they will be indented further because they are contained within folders, not appearing in the TOC at the top level.

```
If docTemp.h_URLpointer(0) <> "" Then  'it is a link
    sMainText = sMainText & "<a href=" & docTemp.h_URLpointer(0) & ">"
Else
    sMainText = sMainText & "<a href=../../h_Index/" & Cstr(docTemp.UniversalID)
        & "?OpenDocument>"
End If
    sMainText = sMainText & sPageSubImg & sStyleTag & docTemp.h_Name(0) & sEndStyleTag
    sMainText = sMainText & sDetailStyleTag
    sMainText = sMainText & " (Last Changed: " & Cstr(docTemp.LastModified)
    sMainText = sMainText & " Size: "
    If docTemp.Size < 1024 Then
        sMainText = sMainText & "0"
    End If
        sMainText = sMainText & Cstr( docTemp.Size / 1024 ) & "k"
        sMainText = sMainText & ")" & sEndStyleTag & "</a>"
```

Update the counter for displaying in the log:

```
nLinkCount = nLinkCount + 1
```

Start the loop again:

```
            End If
        Set docTemp = viewCurrent.GetNextDocument( docTemp )
```

```
    Wend
End If
```

Here we handle the other document types. We have put this code in here because even though it is not relevant in this situation, it provides an easy way to implement mapping of room links.

It would be possible to expand on this and go into the database for the room referenced here, and show all documents in that room. The reason we did not put that in here is because it would breach security principles to allow users to see all room links. If a user did not have access to a room, the link would prompt a user name and password box to appear, and they would be stopped. Users would be informed that a room existed, but they are not able to enter. We have followed the QuickPlace model of only displaying valid links:

```
            Else
            'it must be some other sort of link, such as a room (h_Type = "3")
        End If
    Set docInTOC = viewTOC.GetNextDocument( docInTOC )
Wend
```

Write another line to the log, and finish off the text for the page:

```
Call logAgent.LogAction("Agent Mapped " & Cstr( nLinkCount ) & " documents" )
sMainText = sMainText & "<br>" & sHTML02 & "<br>"
```

This is the final step: output of the accumulated HTML for the page. When we were developing this code, we switched to debug mode (iDebug = 1)and wrote the file to an HTML file on the server. In a production environment ,you will not have the ability to do this, as these functions are disallowed in PlaceBots.

On our internal test server, we have reduced security to allow this to work. To reduce the security, we went into the copy of the database and set the default access to Manager. This is only for the copy of the database used for testing on our test server.

```
    If iDebug Then
        'debug version writes to a file
        fileNum% = Freefile()
        Open "D:\trash\trash.htm" For Output As fileNum%
        Print #fileNum%, sHTMLHead ; sMainText
        Close fileNum%
    Else
        'release version writes to a QuickPlace Main.nsf db
        docReport.PageBody = sMainText
        Call docReport.Save( True, True )
    End If
```

Then we finished off with the exit and error-handing methods:

```
        Exit Sub
```

```
lblLogError:
    Call logAgent.Logerror(Err, Error$)
    Resume Next

End Sub
```

7.4.5 The SiteMap.htm page

The SiteMap.htm page must be manually imported into your Place and is the target document for the PlaceBot. In other words, the Mapperizer PlaceBot goes looking for this file and refreshes it.

It has two parts. The wrapping for the actual map is contained in the HTML file. In the middle of the page is an instruction to render the map, then at the end is more HTML to finish off the page.

Note: The SiteMap PlaceBot and HTML file allow Theme-like customization. In other words, its appearance can be dramatically altered to suit your Place. This is done by altering the HTML in the SiteMap.htm file, editing the images Folder.gif, Page.gif, PageSub.gif and Indent.gif, which are referenced in the SiteMap.htm file.

7.4.5.1 Renaming the SiteMap Page

To rename the SiteMap page, you need to edit a variable in the Mapperizer placebot sMapDocName. When you create a new site map page, rename that with exactly the same name.

If you rename the “SiteMap” page, do not use spaces in the new name. Spaces will cause problems in certain contexts in QuickPlace. There are ways of working around these problems created by adding spaces. However, these can be time-consuming and may cause errors if the core code of QuickPlace changes in future releases.

When referencing the site map page where you have used spaces via a URL, remember to replace the spaces with plus (+) signs.

7.4.5.2 SiteMap HTML Page description

The following section describes the SiteMap HTML page which is imported into your Place and named SiteMap.

It is basically a wrapper for rendering the images on screen and presenting a single JavaScript document.write command. Everything else is either cosmetic presentation or error checking.

The essential lines in this page render the four images on the screen, forcing QuickPlace to upload them:

```
<img src="Folder.gif">
<img src="Page.gif">
<img src="PageSub.gif">
<img src="Indent.gif">
```

The JavaScript command to print the text created by the Mapperizer PlaceBot onto the screen is as follows:

```
<script language=JavaScript>document.write( PageBody )</script>
```

To make the map more presentable, we render the images on the screen in a table. Using a table allows us to create a little vertical space, and also to align the images to the right. This is done to make the pixel images a little less obvious to the reader. Due to the fact that the table is only one pixel high, it can be filled with a color to create a really pretty line.

```
<table width=100% border=0 width=10 cellpadding=0 cellspacing=0>
   <tr>
      <td height=1 align=right><img src="Folder.gif" width="1" height="1"><img
src="Page.gif" width="1" height="1"><img src="PageSub.gif" width="1" height="1"><img
src="Indent.gif" width="1" height="1"></td>
   </tr>
</table>
```

Next comes the section which draws the body of the document onto the page. The first JavaScript Line here makes sure that the PageBody variable has been declared.

Note: If you create content in the PageBody field of a Quickplace document, it is automatically placed into the PageBody JavaScript variable.

If the “typeof” test in the following text determines that the PlaceBot has not run, a message is printed onto the screen giving information on how to get started. This is especially important because it helps avoid a JavaScript error through an undefined variable.

The variable PageBodyMessage is the text you should change if you want to change the message displayed on screen. If you change this text, remember to escape any quotation characters. In other words, put a backslash in front of single or double quotes.

```
<script language=JavaScript>
   if ( typeof( PageBody ) == "undefined" ) {
   var PageBodyMessage = 'Run the Mapperizer Placebot to see a site map .';
      document.write( PageBodyMessage )
   } else{
      document.write( PageBody )
   }
</script>
```

In our SiteMap page, we decided not to put any other HTML at the end of the page. If you want to put some in, it should be located after the </script> tag.

If you have trouble getting the map to appear in the SiteMap document, run the PlaceBot manually and read the log file carefully. When you run the PlaceBot manually, a message will appear telling you that the agent is normally run on a page that is being published....`Do you want to run it anyway?` Just reply `OK` to this. The agent will run fine.

The Log should show the following text.

```
Started running agent 'Mapperizer' on 09/05/2000 09:31:31 PM
09/05/2000 09:31:31 PM: Mapperizer: IBM Redbook Example
09/05/2000 09:31:31 PM: Mapperizer: Run agent: Mapperizer
09/05/2000 09:31:32 PM: Agent Mapped 18 documents
Ran LotusScript code
Done running agent 'Mapperizer' on 09/05/2000 09:31:32 PM
```

If an error occurs, messages like the following will be printed into the window.

```
If you have not loaded the SiteMap page the following error will appear:
Started running agent 'Mapperizer' on 09/20/2000 12:33:49 AM
09/20/2000 12:33:49 AM: Mapperizer: IBM Redbook Example
09/20/2000 12:33:49 AM: Mapperizer: Run agent: Mapperizer
09/20/2000 12:33:49 AM: Mapperizer: Document SiteMap not found
Ran LotusScript code
Done running agent 'Mapperizer' on 09/20/2000 12:33:49 AM
```

Before we move on to our next example, keep in mind the following. Many things have been illustrated with this PlaceBot; we've looked at both how to get deep into the internals of the QuickPlace objects, and at the same time emphasized how to produce good-looking output (which is why a HTML page, a Style Sheet and GIF files are included). However, in many cases where you don't need the most appealing graphic output, you can get the same functionality by just using a PlaceBot.

7.5 The QuickPlace Mail Room Attendant

This example will show you how to add new functionality to Places that will handle your incoming mail using a PlaceBot and QuickPlace Forms. When e-mail is sent to a Place, everything is being put in one single room. Our example consists of a LotusScript PlaceBot and a Form. The Form is used to create configuration documents that link a certain word or phrase in the subject of the e-mail with a folder in the Place.

We call the PlaceBot for the Mail Room Attendant. It is great for handling automatic mail such as news feeds because it relieves you of having to sort the bulk of that mail yourself. For our example, we are using the Mail Room Attendant to sort incoming mail from a list server or news feed into category folders in a room called News Feeds. We work with the following three categories: Hot Topics, News Watch and Tech Today.

The full LotusScript PlaceBot can be download from the IBM Redbooks Web site. See Appendix L, “Additional Web material” on page 425 for information on how to get it.

Before we look at the code, we describe how to install our example so you can try it yourself as you read along.

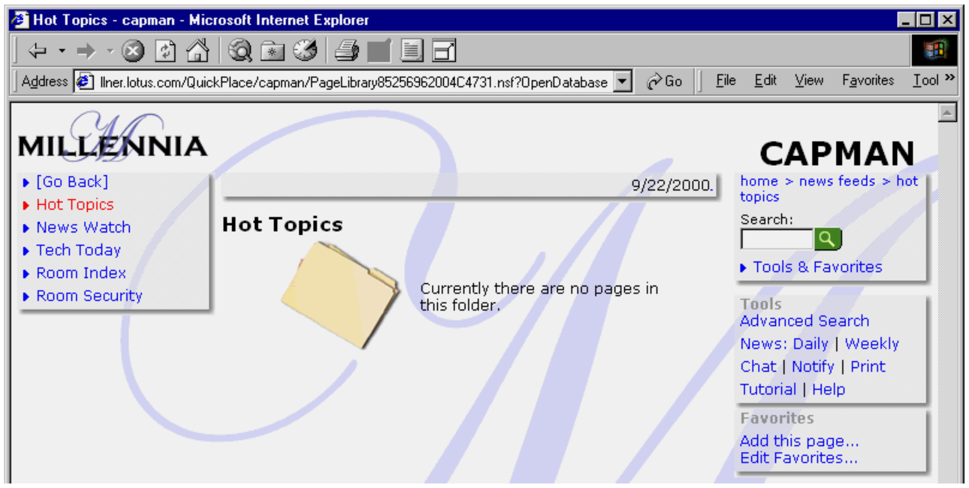
7.5.1 Building the example

The first step is to add two rooms to your Place. Name one Mail Room and place it in the TOC where you like. (We put our Mail Room above Index.) Name the other room News Feeds and put this room above Mail Room in the TOC. Once the rooms are created, delete the Instructions page in each room so the Room Index is the first folder you see and they start out empty.

Then create three folders in the News Feeds room called:

- Hot Topics
- News Watch
- Tech Today

Refer to Figure 71 on page 188.


Hot Topics - capman - Microsoft Internet Explorer

Address: lner.lotus.com/QuickPlace/capman/PageLibrary85256962004C4731.nsf?OpenDatabase

MILLENNIA

CAPMAN

- [Go Back]
- Hot Topics
- News Watch
- Tech Today
- Room Index
- Room Security

9/22/2000.

Hot Topics

Currently there are no pages in this folder.

home > news feeds > hot topics

Search:

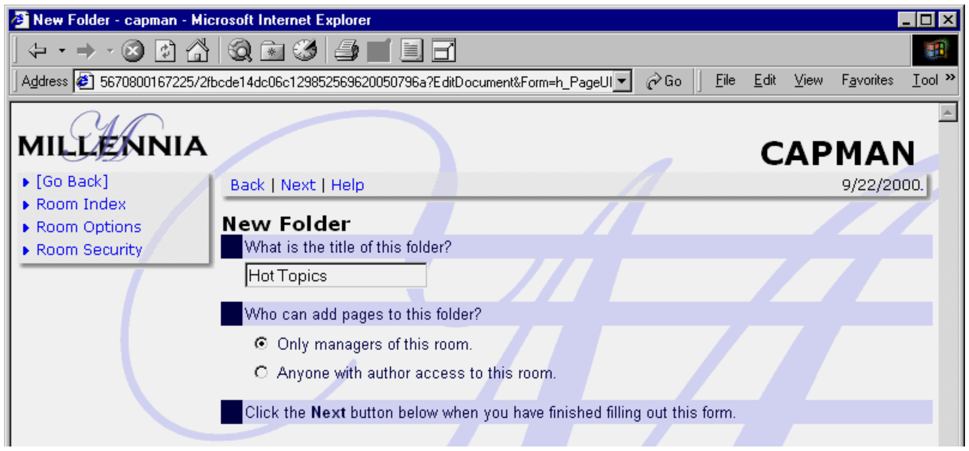
- Tools & Favorites

Tools
Advanced Search
News: Daily | Weekly
Chat | Notify | Print
Tutorial | Help

Favorites
Add this page...
Edit Favorites...

Figure 71. News Feeds room with category folders

We used the standard list presentation for all three folders and set each so only Managers can add pages to them; refer to Figure 72.


New Folder - capman - Microsoft Internet Explorer

Address: 5670800167225/2fbcde14dc06c1298525696200507969?EditDocument&Form=h_PageUI

MILLENNIA

CAPMAN

- [Go Back]
- Room Index
- Room Options
- Room Security

Back | Next | Help

9/22/2000.

New Folder

What is the title of this folder?

Hot Topics

Who can add pages to this folder?

[✓ Only managers of this room.]

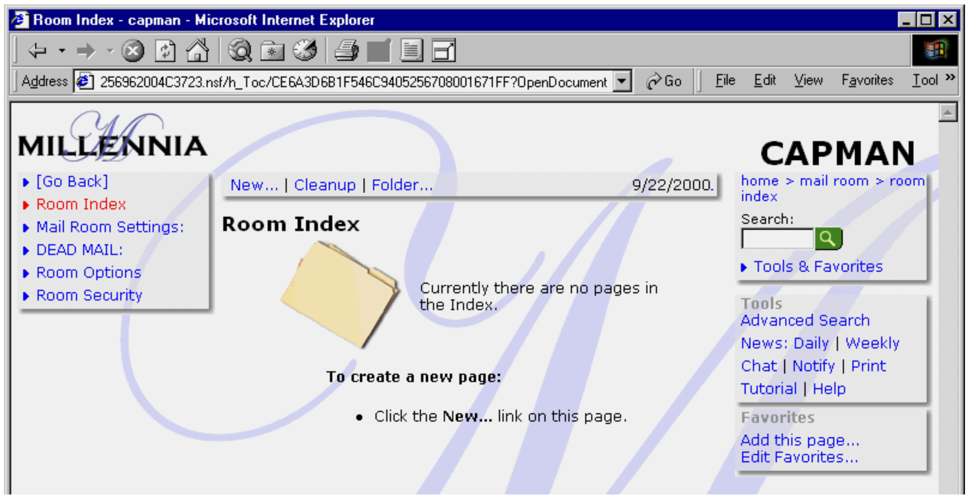
Anyone with author access to this room.

Click the **Next** button below when you have finished filling out this form.

Figure 72. News Feeds Folder with only Managers can create pages setting

7.5.2 Setting up the Mail Room

Now you will need to create two more rooms in the Mail Room. Name the first Mail Room Settings and the second DEAD MAIL. You should delete the Instructions page in each of these new rooms, as we did previously. Mail Room Settings will be used for Mail Room Configuration documents, and DEAD MAIL will be for mail that the Mail Room Attendant cannot determine where to distribute to. Place Managers should check this room periodically for “dead mail” and manually distribute it to its final destination, or they could assign the task to someone else by giving that member access to the Mail Room and its inner rooms.



Room Index

Currently there are no pages in the Index.

To create a new page:

- Click the New... link on this page.

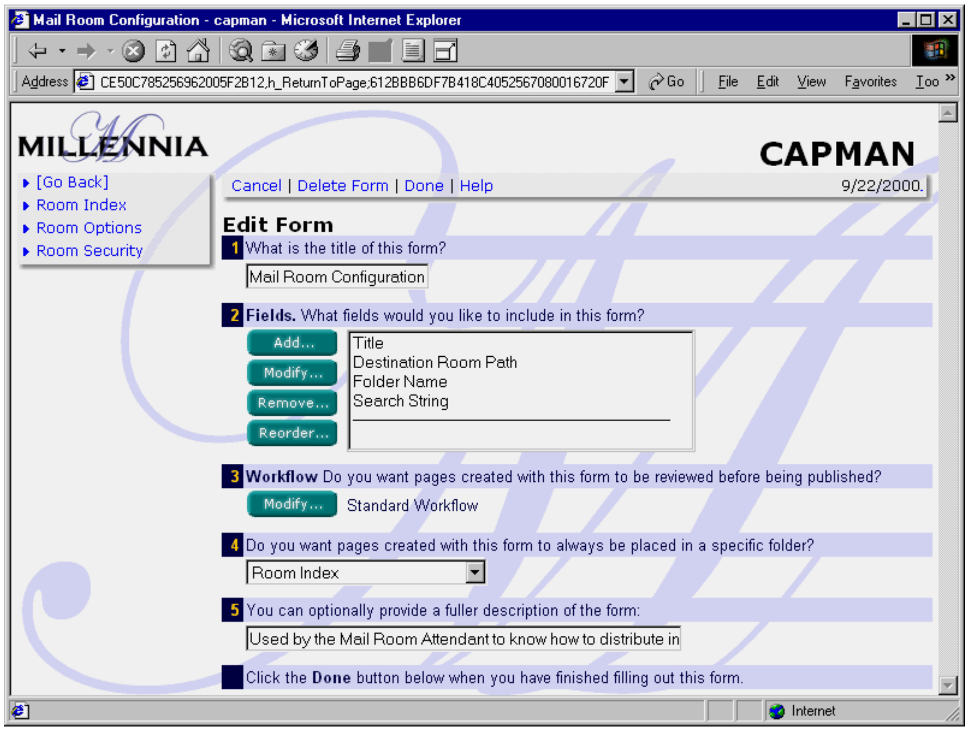
Figure 73. Mail Room showing new Mail Room Settings and DEAD MAIL rooms

Now create a custom Mail Room Configuration Form. This Form will be used to create configurations for the Mail Room Attendant PlaceBot regarding how to distribute incoming mail. To make this Form, go into the Mail Room Settings room and click **Room Options, and** then **New Form**. Select **Simple Form** and click **Next**. Name the Form as follows:

```
Mail Room Configuration
```

In addition to the existing Title field, add these three new Plain Text fields:

- Destination Room Path
- Folder Name
- Search String


Mail Room Configuration - capman - Microsoft Internet Explorer
Address
CE50C785256962005F2B12.h_ReturnToPage;612BBB6DF7B418C40525670800167 20F
Go
File
Edit
View
Favorites
Too
MILLENNIA
CAPMAN
[Go Back]
Room Index
Room Options
Room Security
Cancel | Delete Form | Done | Help
9/22/2000
Add...
Modify...
Remove...
Reorder...
Modify...
Internet

Edit Form

1 What is the title of this form?

Mail Room Configuration

2 **Fields.** What fields would you like to include in this form?

Title
Destination Room Path
Folder Name
Search String

3 **Workflow** Do you want pages created with this form to be reviewed before being published?

Standard Workflow

4 Do you want pages created with this form to always be placed in a specific folder?

Room Index

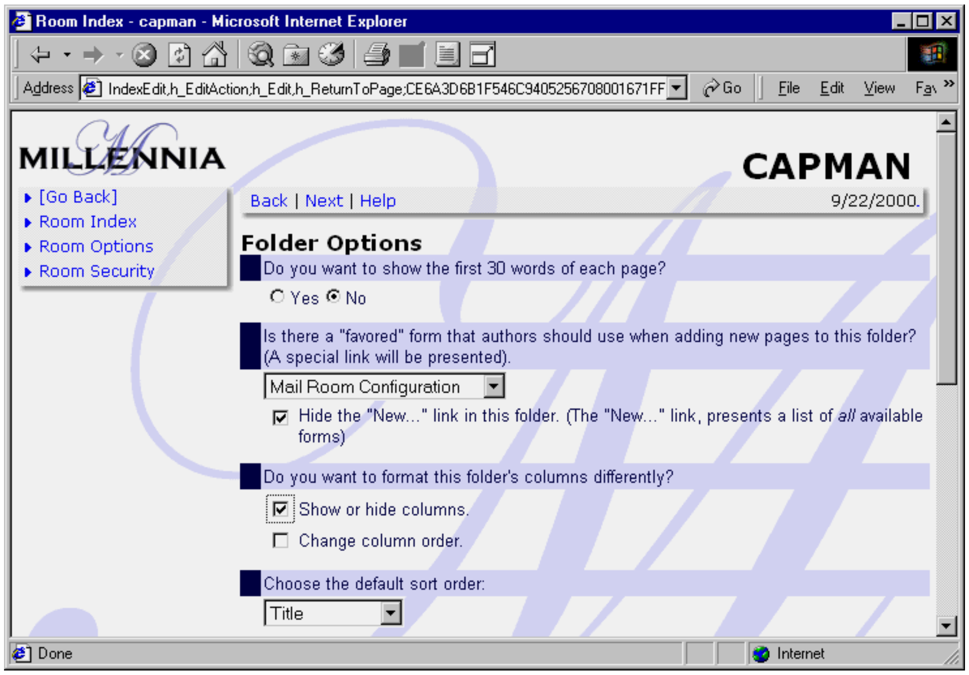
5 You can optionally provide a fuller description of the form:

Used by the Mail Room Attendant to know how to distribute in

Click the **Done** button below when you have finished filling out this form.

Figure 74. New Mail Room Configuration Form

Set the **Room Index** as the folder that you want pages that are created with this Form to always be placed in. Click **Done** and then set the Folder Options of the Room Index as shown in Figure 75 on page 191.


Room Index - capman - Microsoft Internet Explorer

Address: IndexEdit,h_EditAction;h_Edit,h_ReturnToPage;CE6A3D6B1F546C9405256708001671FF

MILLENNIA

CAPMAN

[Go Back]
Room Index
Room Options
Room Security

Back | Next | Help

9/22/2000.

Folder Options

Do you want to show the first 30 words of each page?

Yes [✓ No]

Is there a "favored" form that authors should use when adding new pages to this folder? (A special link will be presented).

Mail Room Configuration

[✓] Hide the "New..." link in this folder. (The "New..." link, presents a list of *all* available forms)

Do you want to format this folder's columns differently?

[✓] Show or hide columns.

[ ] Change column order.

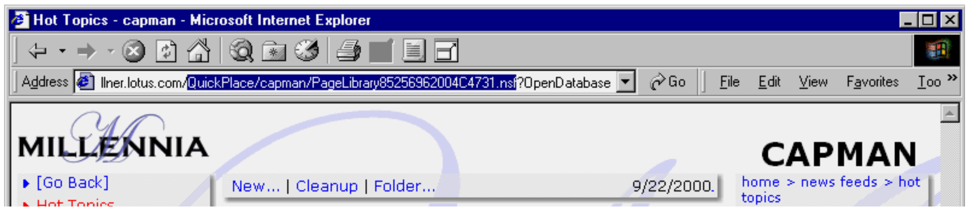
Choose the default sort order:

Title

Done

Figure 75. Mail Room Settings - Room Index folder options

Click **Next** and uncheck all columns but *Title*. Click **Next**, and then click **Next** again to leave the folder where it is. Navigate to the **News Feeds** room and copy from the URL in the browser's address box the portion, as shown in Figure 76.


Hot Topics - capman - Microsoft Internet Explorer

Address: llner.lotus.com/QuickPlace/capman/PageLibrary85256962004C4731.nsf?OpenDatabase

MILLENNIA

CAPMAN

[Go Back]

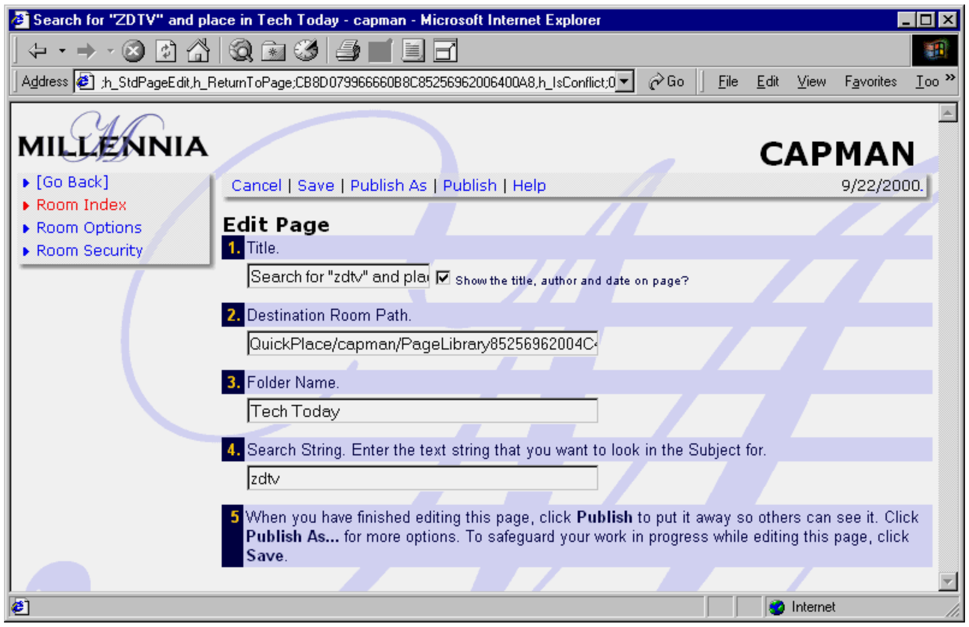
New... | Cleanup | Folder...

9/22/2000.

home > news feeds > hot topics

Figure 76. Copy text for Destination Room Path from URL in browser's address box

Now create a New Mail Room Configuration, as shown in Figure 77 on page 192, using the URL text you just copied for the Destination Room Path field.


Search for "ZDTV" and place in Tech Today - capman - Microsoft Internet Explorer

Address :h_StdPageEdit,h_ReturnToPage;CB8D0799666 60B8C85256962006400A8,h_IsConflict;0

MILLENNIA

CAPMAN

- [Go Back]
- Room Index
- Room Options
- Room Security

Cancel | Save | Publish As | Publish | Help 9/22/2000.

Edit Page

1. Title.

Search for "zdtv" and pla ☑ Show the title, author and date on page?

2. Destination Room Path.

QuickPlace/capman/PageLibrary85256962004C

3. Folder Name.

Tech Today

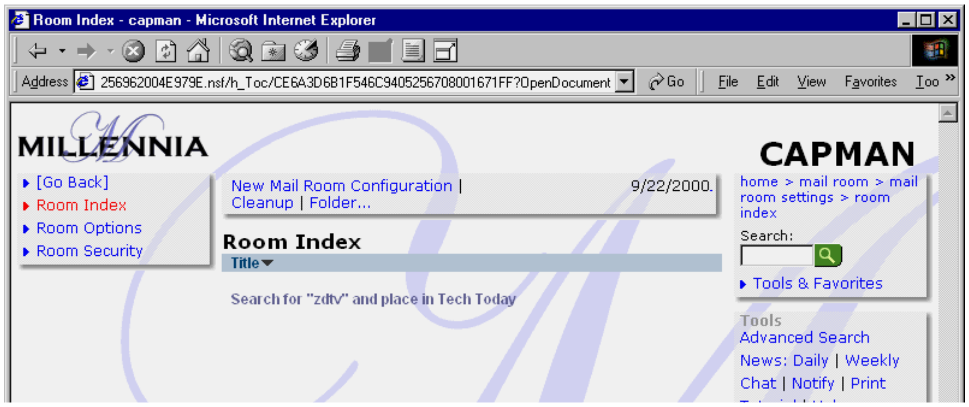
4. Search String. Enter the text string that you want to look in the Subject for.

zdtv

5 When you have finished editing this page, click **Publish** to put it away so others can see it. Click **Publish As...** for more options. To safeguard your work in progress while editing this page, click **Save**.

Figure 77. Create Mail Room Configuration document

Figure 78 shows the Mail Room Settings room with the new Configuration document we just created.


Room Index - capman - Microsoft Internet Explorer

Address 256962004E979E.nsf/h_Toc/CE6A3D6B1F546C9405256708001671FF?OpenDocument

MILLENNIA

CAPMAN

- [Go Back]
- Room Index
- Room Options
- Room Security

New Mail Room Configuration | Cleanup | Folder... 9/22/2000.

home > mail room > mail room settings > room index

Room Index

Title

Search for "zdtv" and place in Tech Today

Search:

- Tools & Favorites

Tools

Advanced Search

News: Daily | Weekly

Chat | Notify | Print

Figure 78. Mail Room Settings room with new Configuration document

7.5.2.1 Writing the MailRoomAtendant.lss PlaceBot

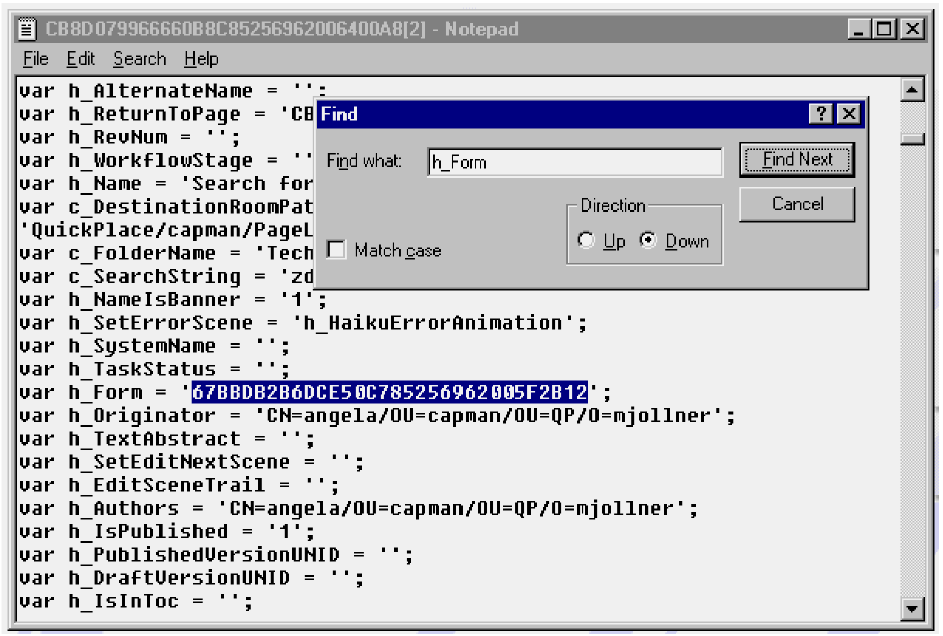
The full LotusScript PlaceBot can be downloaded from the IBM Redbooks Web site. See Appendix L, “Additional Web material” on page 425 for information on how to get it.

The LotusScript PlaceBot does the following:

- It reads all configuration documents in the Mail Room Setting room.
- It finds all new documents in the Mail Room.
- It compares the subject of the new mail with its configuration documents.
- If there is a match, it moves the mail to the specified folder; otherwise, the mail is moved to the DEAD MAIL room.

In the following sections, we describe the modifications you need to make to this file to make it work in your Place or QuickPlace PlaceType.

We need to make three changes to the PlaceBot. In order to make them, we have to get some information from the additions we have made so far. First, we get the *h_Form* property value from the Mail Room Configuration Form we just built. To do this, we simply open the configuration document that we created and view the HTML source of this page. We do a search for h_Form and copy the text value, as shown in Figure 79 on page 194.



```
var h_AlternateName = '';
var h_ReturnToPage = 'CB
var h_RevNum = '';
var h_WorkflowStage = ''
var h_Name = 'Search for
var c_DestinationRoomPat
'QuickPlace/capman/PageL
var c_FolderName = 'Tech
var c_SearchString = 'zd
var h_NameIsBanner = '1';
var h_SetErrorScene = 'h_HaikuErrorAnimation';
var h_SystemName = '';
var h_TaskStatus = '';
var h_Form = '67BBDB2B6DCE50C785256962005F2B12';
var h_Originator = 'CN=angela/OU=capman/OU=QP/O=mjollner';
var h_TextAbstract = '';
var h_SetEditNextScene = '';
var h_EditSceneTrail = '';
var h_Authors = 'CN=angela/OU=capman/OU=QP/O=mjollner';
var h_IsPublished = '1';
var h_PublishedVersionUNID = '';
var h_DraftVersionUNID = '';
var h_IsInToc = '';
```

Figure 79. Getting the h_Form property value of the Mail Room Configuration form

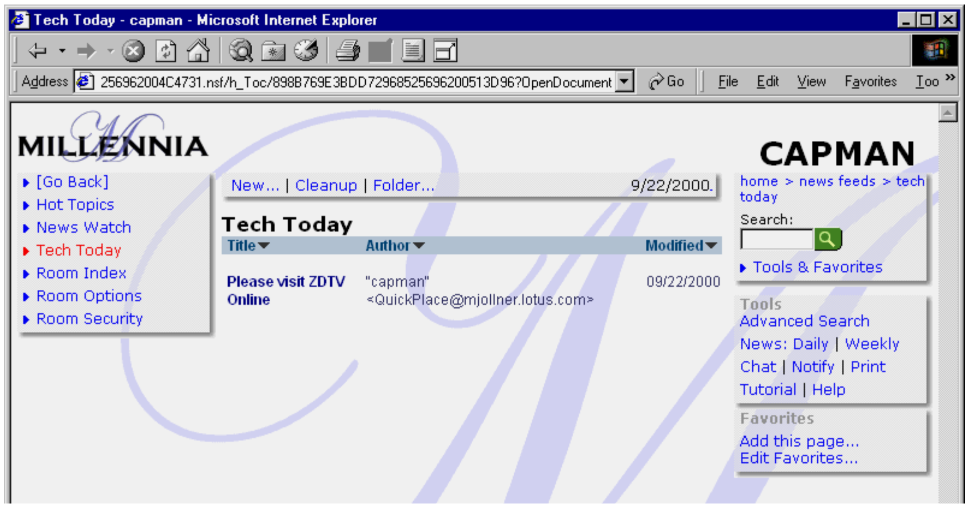
Next we need to get the text from the URL for the Mail Room Settings room, as we did for the News Feeds room. The last text value we need to get is the text from the URL for the DEAD MAIL room. Once we have these three values, we are ready to create our PlaceBot.

Create a new PlaceBot in the Mail Room and call it Mail Room Attendant. Choose to have it run on a **schedule**, and set the PlaceBot to affect **All pages** in the room that is in the Room Index. For testing purposes, set the PlaceBot to run every 5 minutes. Click **next** and then attach the MailRoomAttendant.lss file. Open the file to add the text values that we collected previously. Add the text values to the three constants defined at the top of the script. Close and save the script. Click **Done** when you are ready to save your PlaceBot.

The last change we need to make is to set the Incoming Mail Setting to have mail come to the Mail Room. To do this, navigate to the Main room and click **Customize**. Click **Basics** and then **Change Basics**. Scroll down and select **Yes - into room Mail Room**.

To test the PlaceBot, send a mail message to your Place with the phrase *zdtv* somewhere in the Subject. The mail message should be distributed to the

Tech Today folder in the News Feeds room, based on the Mail Configuration document that we created. Figure 80 shows the result of our successful Mail Room Attendant test.



Tech Today

Title	Author	Modified
Please visit ZDTV Online	"capman" <QuickPlace@mjollner.lotus.com>	09/22/2000

Figure 80. Successful test

We now look at an example of how you can add functionality to your QuickPlace by using Domino agents.

7.6 Using Domino agents with QuickPlace

If you have QuickPlace installed as an overlay on top of a Domino server, you can write Domino agents that reside in Domino databases to enhance the functionality of Places, integrate between Places and Domino applications, write utilities that help administer the QuickPlace server, and so on.

In 10.1, “Mirroring data to Notes - a Java agent example” on page 277 we show how to get a QuickPlace calendar to work together with a calendar in a Domino database, and in the following section we describe how you can implement a useful tool for QuickPlace administrators by having a scheduled agent run in a Domino database.

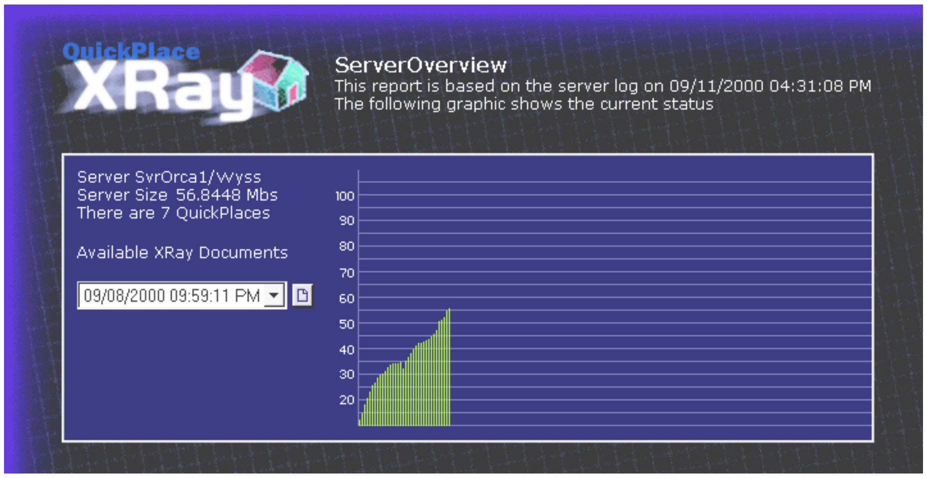
7.6.1 TheXRay server status reporter agent

We call our sample application TheXRay. Its purpose is to create a graphical overview record of the contents of a server via a browser. We want to see server statistics such as number of Places on the server, their age and use, the disk volume used by the Places, and so on.

TheXRay can be downloaded from the IBM Redbooks Web site. The following description will be easier to follow is you have the sample installed. See Appendix L, “Additional Web material” on page 425 for information on how to get the sample.

Note: TheXRay is just a sample used to illustrate programming techniques and not may not work in all environments. For example, if your Places have a lot of inner rooms TheXRay may not work correctly.

TheXRay needs to be installed on a Domino or QuickPlace server, but apart from the installation, all access is via the Web. When you first enter the database, you will see an overview page that looks something like Figure 81..



ServerOverview
This report is based on the server log on 09/11/2000 04:31:08 PM
The following graphic shows the current status

Server SvrOrca1/Wyss
Server Size 56.8448 Mbs
There are 7 QuickPlaces

Available XRay Documents

09/08/2000 09:59:11 PM

Figure 81. The overview page

This overview page has the logo of the application in the top left corner, and near that is the heading, showing the date of the last time that statistics were collected on the server.

In the left-hand side of the blue box are more statistics about the server. The Server name is listed first, below that is the current Server Size, followed by the number of Places on the server.

The drop-down list on the left-hand side enables you to choose an XRay document to view; refer to Figure 82. It displays a number of dates, each one representing a different XRay document.

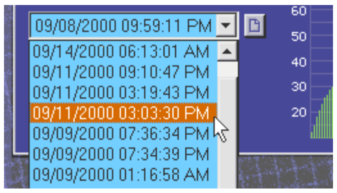


Figure 82. The list of existing XRay documents

The Go button has an image of a page on it; it will take you to whichever page you select in the drop-down field.

A very hip feature of TheXRay is the ability to view the most recent record made of the server. By selecting the Most Recent option from the drop-down list and pressing Go, a special Domino URL is called and you are taken to the most up-to-date record contained in the database; refer to Figure 83.

Figure 83. The Most Recent option

Referring back to the overview page shown in Figure 81 on page 196 , on the right-hand side of the blue box is a graph showing the volume of the Places on the server. Each green bar represents a different XRay document. It is recommended that you run TheXRay agent *every day on a scheduled basis*. This way you can get a clear picture of the server's growth using this graph. If TheXRay agent is triggered manually, the time between each survey will not be constant and rate of change displayed in the graph will be misleading.

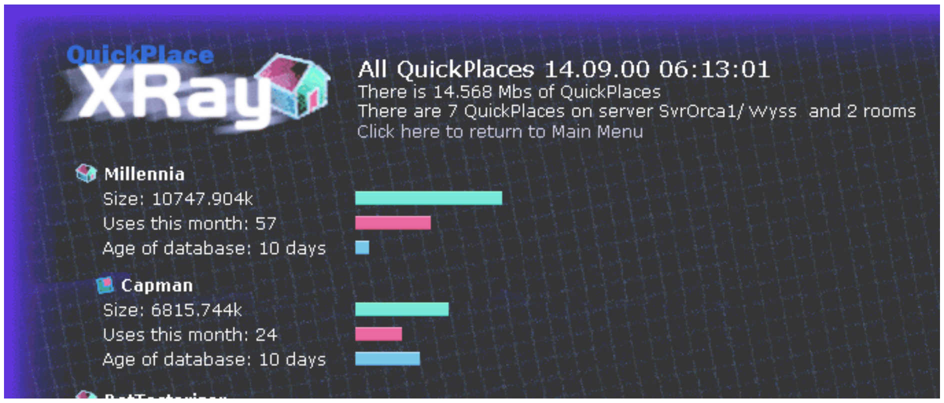
7.6.2 Manually triggering the agent

TheXRay database and agent are set up to run on a scheduled basis. If you want to run the agent "on the fly", you can do it by running the agent via the "runTheXRay" agent.

To avoid having TheXRay agent create a new document each time you run the agent manually, you must change a few lines in the agent. The agent can then be run from a browser, and the agent will print all of the HTML to the screen. Because the entire document is rendered as one stream of HTML, you can just redirect this to a browser window and it renders in the same way as if you had created a scheduled XRay document.

To make these changes, build a Form to prompt the agent, and in the Bottom of the initialize event, force the agent to print all of the HTML that it was saving to a document. This will write the HTML to the browser window.

If you click the Go button, you will be able to see the document that you have selected. When we ran TheXRay on our server, it looked as shown in Figure 84.



All QuickPlaces 14.09.00 06:13:01
There is 14.568 Mbs of QuickPlaces
There are 7 QuickPlaces on server SvrOrca1/Wyss and 2 rooms
Click here to return to Main Menu

Millennia
Size: 10747.904k
Uses this month: 57
Age of database: 10 days

Capman
Size: 6815.744k
Uses this month: 24
Age of database: 10 days

Figure 84. Server record showing the Millennia Place and the Capman Room

The server record has TheXRay logo at the top, which provides a link to the overview page. To the right of the logo we see some statistics about the Places on the server. The top line shows the date and time the record was created. Then the server volume appears, followed by the number of Places and the number of Rooms belonging to the Places. The final line is a link back to the overview page.

In the main body of the page, statistics are displayed. You can see here the Millennia Place, which was about 10.7 MB. It had 57 uses this month, and it was 10 days old. The icon of the house denotes that “Millennia” is a Place.

Below the Millennia entry we see an icon of a Room (a blue square inside another blue square, with a red center). This icon denotes that “Capman” is a Room. Subrooms are represented in the same way as Places.

7.6.3 Components

The application has a few major functional components, as shown in Table 29.

Table 29. Components of TheXRay application

Component	Type	Description
vwSizeGraph	View	Used to display a graph of the size of the server. It is the first page users see when they open the database. When the vwSizeGraph view is opened, they also have the ability to open any of the XRay documents.
fmXRay	Form	The Domino Form used to organize the information created by the agent. This is a simple form with only 3 visible fields. The first field is a reference to the database path for creating URLs and JavaScripts. The next field prevents the form from being saved. The PageBody field holds the HTML for the page.
TheXRay	Agent	The Domino agent that creates a snapshot of the server. This agent is described in full in the following pages.

7.6.3.1 How TheXRay agent works

TheXRay agent works as follows:

1. It initializes the HTML for the page.
2. It looks for a Place.
3. It gets all the rooms for the current Place by using subGetPlace.
4. It gets the database's stats from the Log.nsf by using fnGetDbStats.
5. It determines if the Place has any more Rooms by using subGetRoom.
6. It gets the database's stats from the Log.nsf by using fnGetDbStats.
7. It creates header report HTML.
8. It concatenates all HTML text.
9. It creates a new Domino database document with the results.

7.6.3.2 A note about security

This agent can be run on the server without having been signed by the Server ID. This has the potential to allow an administrator to see the contents of Places, without actually opening them. TheXRay agent does not require access to Places, but can still get low security information about them by accessing the server Statistics Log (Log.nsf).

This method of gathering statistics adds an extra level of complexity, but we can avoid the political issues of accessing customer data. This is relevant in an ASP situation where there may be sensitive information on the server.

Another alternative would be to create an agent which opens each of the databases and retrieves dates and figures directly. This has the potential of providing more detailed, up-to-date information. Doing this is slightly simpler than the way TheXRay works, and could use the code provided in the Mapperizer PlaceBot section.

For more information about this, see 7.1.2, “PlaceBot security” on page 167.

7.6.3.3 Option declarations

The agent starts with some declares to make sure that the variables have been declared before they are used. This is done in the Option Declare line and helps to avoid spelling mistakes, because an error will occur if you try to save the agent with an undeclared variable.

```
Option Public
Option Declare
```

7.6.3.4 Global declarations

These variables are used in more than one of the functions. For this reason, we declare them in the global declarations. All global variables begin with g_ to indicate that they are global. There is more information on variable naming conventions in Appendix E, “Naming convention used” on page 379.

```
Dim g_ses As NotesSession
Dim g_ndbServerLog As NotesDatabase
Dim g_vwLogDbUsage As NotesView
Dim g_dclLogEntry As NotesDocumentCollection
Dim g_sHTML As String
Dim g_sDbSizeText As String
Dim g_sDbManager As String
Dim g_sDbMonthUsesText As String
Dim g_sDbAgeText As String
Dim g_longServerSize As Long
Dim g_sURLTemp As String
Dim g_sServerName As String
Dim g_longServerSizeDisplayUnits As Long
Dim g_longMonthUsesDisplayUnits As Long
Dim g_sFolderName As String
Dim g_iRoomCount As Integer
Dim g_iPlaceCount As Integer
```

7.6.3.5 Initialize

The agent begins the main body. The Initialize event provides the central thread for the application. It starts off declaring a number of variables required for use only within the Initialize event:

```
Sub Initialize
%REM /*--------=====oooOoo===-----
```

```
    TheXRay
    Description: cycles through directory structure on a server,
        finding all files in the QuickPlace folder which are
        either QuickPlaces or rooms.
        Due to ACL concerns, statistics are retrieved from the
        servers log.nsf.
        By signing the agent with the administrators id, the
        agent can be changed to also look into the QuickPlaces.
    Possible future changes:
        Display results based on a skin file
%END REM-------===oooOoo===----


'//declare local variables
    Dim ndbThis As NotesDatabase
    Dim docReport As NotesDocument
    Dim dtQuery As New NotesDateTime( "" )
    Dim vwSizeGraph As NotesView
    Dim sQuery As String
    Dim sHTMLHead As String
    Dim sHTMLTitle As String
    Dim sNdbRoomSize As String
    Dim iFileNum As Integer
    Dim iDebug As Integer
```

Then it starts to set those variables, creating a session and finding the database.

```
'//set global variables
    Set g_ses = New NotesSession
    g_sServerName = "SvrOrca1/Orca"
```

In a default Domino setup, the Log.nsf is in the root directory of the server, and is named Log.nsf. *We strongly advise that this database not be moved*.

```
    Set g_ndbServerLog = New NotesDatabase( g_sServerName , "log.nsf")
    Set g_vwLogDbUsage = g_ndbServerLog.GetView( "DatabaseUsage" )
```

Next the code creates a query string and assigns it to the variable *sQuery*, The query string is somewhat like a Domino view Selection formula.

View selection formulas

When you design an application, there are times when you want to access all documents and times when you want to access only certain documents in the database. Most databases have one view that shows all documents and other views that show a subset of documents.

For example, if you just wanted to access documents where the field named *Form* has the value *fmReport*, you use the following selection formula:

```
SELECT Form = "fmReport"
```

The default selection for new views is the combination of the SELECT statement and the @All formula: `SELECT @All`, which means “include all database documents in this view.”

To narrow down the kinds of documents the view displays, add a formula that displays only particular documents. For example, in a task-tracking database, the Work in Progress view could display only those documents whose Status field does not contain “Complete.” In a brainstorming database, the Design Ideas view could display all documents categorized as “Design Suggestion.” Here is a formula that displays only documents containing new features and displays them in a What's New view: `SELECT @IsMember("01 What's new?"; View)mber("01 What's new?"; View)`

Here is the code for the query string and its execution when building the document library, which is stored in the global variable g_dclLogEntry.

```
    sQuery = |Form = "Activity" & @left( @lowercase( Pathname ) ; "\\" ) = "quickplace"|_
    & | & @rightback( @lowercase( Pathname ) ; "\\" ) != "search.nsf"|
'//GetAllDocumentsByKey wouldn't work
    Set g_dclLogEntry = g_ndbServerLog.Search( sQuery , dtQuery , 0 )
```

To understand this query string or selection formula you need to look one line past it, to where the g_dclLogEntry variable is set. “g_” tells you that the variable is global and “dcl” establishes that it is a document collection. Using the sQuery string we can create a list of databases that the agent needs to look at. Remember that this agent does not look at non-QuickPlace databases.

Is it a Place database or a Room database

The way we know whether a database is a Place database or a Room database is via two pieces of information. This agent only looks for Places and Rooms. Places are always in the \Quickplace\ directory in directories bearing the name of the Place. For example, the Place Millennia has the following path:

```
\Quickplace\Millennia\Main.nsf
```

Note: All QuickPlaces have the file name Main.nsf. The name of the Place can be viewed by the folder name.

Rooms are always have a structure something like PageLibrary0123456789ABCDEF.nsf. Rooms always start with PageLibrary, and then have a 16-digit hexadecimal time stamp string. This hexadecimal string provides a unique ID for the file. After the ID number, there is always the .nsf ending.

You can tell which Rooms belong to which Places because the Rooms and Places are always in the same folder. This is how a Room belonging to the Millennia Place would look in the directory structure:

```
\Quickplace\Millennia\PageLibraryC3D0083DA008AC3A.nsf
```

To find all our Places, the first thing we need to do is look in the Domino data directory, in the "Quickplace" directory.To make the comparison a little more robust, we compare the two strings in lowercase. In other words, the agent compares "quickplace" and "quickplace" strings.

In every QuickPlace folder, there are other databases that a Place uses (for example, Contacts1.nsf and Search.nsf). When Millennia set up this application, they decided that they did not want to know the size of these helper databases. However, if your client wants this information, it is possible to retrieve these statistics as well.

Note: Another way of doing this would be by using the LotusScript function that can get all documents in a view that matches a key value, but this was more complex due to the structure of the views in the Log.nsf database.

The next few lines set some variables that are used for counting documents, sizes of files, and so on. They must be global because they are updated in a number of the functions and subs in the agent:

```
   g_iPlaceCount = 0
   g_iRoomCount = 0
   g_longServerSize = 0
   g_longMonthUsesDisplayUnits = 10
   g_longServerSizeDisplayUnits = 102400
```

It is time to give a bit of feedback, so we print that the agent has started:

```
    Print( |started agent | & g_ses.CurrentAgent.name)
```

The agent has actually started now and it starts to build the HTML, beginning with the header.

Note: If you use this code in a PlaceBot or agent that creates content in QuickPlace documents, the <head> , </head> tags will be removed by QuickPlace.

To include JavaScript code, put your JavaScript after the <body> tag, in the main part of the document.

The <body> tag will also be removed. Page attributes that you would normally set in the <body> tag can be set via the stylesheet selector named "h-page-bg". For more information on stylesheets this see the section on Themes.

```
    sHTMLHead = |<html>
    <head>
        <link rel=stylesheet type="text/css" href="../StyleSheet.css">
    </head>
    <body class=m-page-bg>
    <table width=100% border=0>|
```

The agent needs to initialize the g_sHTML string here because it is added to in loops in the following Subs and Functions:

```
'//initialize html body string
    g_sHTML = ||
```

Error handling is set up. We recommend that you comment this out if you edit this code and want to test it, because the error handling here will *mask the effects of errors* and make debugging difficult:

```
'//set up error handling turn this off during debugging
    On Error Goto lblLogError
    Dim logAgent As New NotesLog( g_ses.CurrentAgent.name )
    logAgent.LogActions = True
    Call logAgent.OpenAgentLog
    Call logAgent.LogAction("ServerMap: IBM Redbook Example" )
```

This is where the agent branches off to start finding Places and Rooms. First of all the subGetPlace is called. This in turn calls the subGetRoom Sub, to retrieve information about rooms. These subs are commented separately later in this section:

```
'//start Place and Room reporting
   Call subGetPlace
```

After calling the subGetPlace Sub, we have returned. The g_sHTML variable is quite large, containing the body of the document. The size of a string variable is limited only by memory, but we have tried to be as efficient as possible.

Other variables have also now been set, and the agent has counted all of the Places and Rooms. Using these returned variables, we can now build the title. The title differs from the Header, because the Header is HTML text to help the browser initialize the page, and the title builds *readable* HTML:

```
'//build html report title
   sHTMLTitle = |<td width=26 valign=top>
      <img src=../ecblank.gif width=26 height=14 border=0>
   </td>
   <td width=216 valign=top class=m-head-text>
      <img src=../ecblank.gif width=216 height=20 border=0><br><a
href="../../TheXRay.nsf">
   <img src=../logoXRay.gif width=203 height=71 border=0 alt="Click here to see the
overview"></a>
   </td>
   <td width=100% valign=top class=m-head-text>All QuickPlaces | & Cstr(Now()) & |<img
src=../ecblank.gif width=1 height=45><br>|
   sHTMLTitle = sHTMLTitle & "<span class=m-subHead-text>There is " & Cstr( Round(
g_longServerSize / 1024000 , 3 ) ) & " Mbs of QuickPlaces<br>There are " &
Cstr(g_iPlaceCount ) & " QuickPlaces on server "
   sHTMLTitle = sHTMLTitle & g_sServerName & " and " & Cstr(g_iRoomCount)  & "
rooms</span></td></tr>"
   sHTMLTitle = sHTMLTitle & "</table><table width=100% border=0>"
```

The agent now puts all the pieces together in the correct order. This order may seem a little strange, but remember that the report body had to be created before the agent had the information to create the title.

```
'//combine html header, report title and body
   g_sHTML = sHTMLHead & sHTMLTitle & g_sHTML & "</table><br><br>"
```

The agent finishes by either writing to an HTML file on the file directory structure or creating a new document in a Domino database. We recommend that you have the agent write to a Domino database, because then these documents can be analyzed and managed using the functionality of a Domino database.

We recommend that you write to a file when testing because it is quicker to just refresh the HTM document in a browser. If you write to a Domino database, a new document is created each time and you need to find this new document in the browser each time.

Debugging the HTML code is done best by viewing the source of the document. The agent produces a lot of HTML, but due to the fact that the HTML is written in loops by the agent, if there is a mistake it can be found many times in the output file, but you just need to fix it once in the script.

```
'//output report
   iDebug = 0
   If iDebug Then
      'debug version writes to a file
      iFileNum% = Freefile()
      Open "D:\dwyss\TheXRay\TheXRay.htm" For Output As iFileNum%
      Print #iFileNum%, g_sHTML
      Close iFileNum%
   Else
      'write to a notes document in the current db
      Set ndbThis = g_ses.CurrentDatabase
      Set docReport = New NotesDocument( ndbThis )
      docReport.Form = "fmXRay"
      docReport.sQuickPlaceCount = Cstr(g_iPlaceCount )
      docReport.sRoomCount = Cstr( g_iRoomCount )
      docReport.sServer = g_sServerName
      docReport.sSErverSize = g_longServerSize
      docReport.PageBody = g_sHTML
      Call docReport.Save( False, False )
      Set vwSizeGraph = ndbThis.GetView("vwSizeGraph")
      Call vwSizeGraph.Refresh
   End If
```

The function starts its ending procedures, first of all printing out that it has finished, then ending the Sub:

```
   Print( |----- end ------|)

lblLogError:
   Call logAgent.Logerror(Err, Error$)
   Resume Next
End Sub
```

This section of the agent is now finished, and we move on to next function fnGetDbStats, which is used to retrieve statistics from the Statistics Log database.

7.6.3.6 fnGetDbStats Function

This function retrieves statistical data from the Log.nsf database. The Log database is a Domino database containing documents with statistical data about all databases on a server.

The main reason for using functions is to help to break down an agent into smaller chunks that can be reused, but they also make agents easier to understand.

The Domino statistics log Log.nsf is very useful because it contains the information about databases on the server that most administrators want to know, so our agent only needs to have access to this database. Otherwise, the access control lists for all Places on the server need to be modified to allow access to the ID of the agent creator.

It is possible to retrieve all the QuickPlace information we are gathering by looking into the Log. This agent gets some information from the server document collection and some information from Log.nsf.

We chose to split the information gathering into two parts in order to exploit the strengths of the approaches. Looking at the server directly via a server document collection provides an up-to-date list of the Places, and the Log provides server statistics without having to compromise the security of information in Places.

If you use only the Log to gather statistical information, you wouldn't know if a Place existed until the Log is updated. By using the document collection, if a Place was created since the Log was run, we at least know it is there, but the file sizes and so on will not appear.

Note: You can manually trigger the Statlog task to run by opening the server console and typing the following:

```
load statlog
```

The server will start chugging away and the following line will appear within a couple of seconds:

```
24/12/2000 14:34:33 Starting update of database usage statistics
```

The following description of the statlog's functionality describes its default configuration. In its default configuration, a task called statlog runs every night at 05:00 on a Domino server. This task writes a status report of an entire server to special documents in the Log.nsf.

Note: TheXRay agent should be run *after* the statlog task to ensure that the the information TheXRay displays is as up to date as possible. If TheXRay runs one hour before the statlog, then the information displayed in TheXRay database will be 23 hours old.

The Notes.ini indicates to the server when the statlog will run. The default line in the Notes.ini looks as follows:

```
ServerTasksAt5=Statlog
```

We suggest that the Statlog be run at 03:00 so that there is time for the scheduled agent to run later in the morning. To do this, add the string Statlog to the ServerTasksAt3 event. Our server already had a server task listed there called Gogs, so we separated them with a comma.

```
ServerTasksAt3=Gogs,Statlog
```

Note: If you are testing TheXRay on a machine which is turned off at night, the statlog task will not run. We recommend that you trigger the task manually, using the statlog command.

The fnGetDbStats starts off by declaring the variable sNdbTempFilepath as a string, as well as declaring itself as a string. This is so that it can be referenced and return a string on completion.

```
Function fnGetDbStats( sNdbTempFilepath As String ) As String
```

In order to look into Log.nsf, we need to declare variables to help get the necessary files:

```
'declare variables for looking into the serverlog
   Dim docLogEntry As NotesDocument
   Dim docLogEntryNext As NotesDocument
   Dim iDbSize As String
   Dim sDbTextPre As String
   Dim sDbSizeTextMid As String
   Dim sDbMonthUsesTextMid As String
   Dim sDbAgeTextMid As String
   Dim sDbTextPost As String
   Dim sDbDaysOld As String
   Dim longTemp As Long
```

Now the agent starts looping through documents in the document Log, finding relevant ones:

```
   Set docLogEntry = g_dclLogEntry.GetFirstDocument
   Do
      Set docLogEntryNext = g_dclLogEntry.GetNextDocument( docLogEntry )
      If Not  ( docLogEntryNext Is Nothing ) Then
         If (docLogEntry.Pathname(0) = sNdbTempFilepath ) Then
            Exit Do
         Else
            Set docLogEntry = docLogEntryNext
         End If
      Else
         Exit Do
      End If
   Loop

   If (docLogEntry Is Nothing) Then
      'do nothing
   End If
```

Now that the agent has found a document matching some of the necessary criteria, it tests to see if the document represents a valid Place and if not, removes the document from the collection. The reason for doing this is so that our looping process becomes faster and faster with each iteration due to the fact that there are fewer and fewer documents in it.

It is important to understand that the DeleteDocument method does not delete the database, it only deletes *a document referencing it* in the temporary collection.

```
   If docLogEntry.Pathname(0) = sNdbTempFilepath Then Call g_dclLogEntry.DeleteDocument(
docLogEntry )
```

Now that the agent has found a document matching the necessary criteria, it begins setting some default values into variables. These default variables will produce four blank table cells. Two of the cells will include the “unknown” tag:

```
   sDbTextPre = "<tr><td>&nbsp;</td><td width=200> &nbsp; &nbsp; "
   sDbSizeTextMid = "unknown</td><td>&nbsp;"
   sDbMonthUsesTextMid = "unknown</td><td>&nbsp;"
   sDbAgeTextMid = "unknown</td><td>&nbsp;"
   sDbTextPost = "</td></tr>"
   g_sDbManager = ""
```

If we find a document matching the selection criteria, the agent looks into the document and overwrites some of the variables we just declared with data. In other words, if we find the correct data, overwrite the empty table cell with a filled table cell, containing the fomatted data:

```
   If Not (docLogEntry Is Nothing ) Then
       longTemp = docLogEntry.DiskSpace(0)
       g_longServerSize = g_longServerSize + longTemp
       sDbSizeTextMid = Cstr( longTemp / 1024 ) & "k"
       sDbSizeTextMid = sDbSizeTextMid & "</td><td><img src=../graphGreen.gif width="
       sDbSizeTextMid = sDbSizeTextMid & Cstr( Round( longTemp /
g_longServerSizeDisplayUnits , 0 ) )
       sDbSizeTextMid = sDbSizeTextMid & " height=10>"

       longTemp = docLogEntry.MonthUses(0)
       sDbMonthUsesTextMid = Cstr( longTemp )
       sDbMonthUsesTextMid = sDbMonthUsesTextMid & "</td><td><img src=../graphRed.gif
width="
       sDbMonthUsesTextMid = sDbMonthUsesTextMid & Cstr( Fix( longTemp /
g_longMonthUsesDisplayUnits ) )
       sDbMonthUsesTextMid = sDbMonthUsesTextMid & " height=10>"

       sDbDaysOld = Cstr( Clng( Now() ) - Clng( docLogENtry.Created ) + 1 )
       sDbAgeTextMid = sDbDaysOld & " days"
       sDbAgeTextMid = sDbAgeTextMid & "</td><td><img src=../graphBlue.gif width="
       sDbAgeTextMid = sDbAgeTextMid & sDbDaysOld
       sDbAgeTextMid = sDbAgeTextMid & " height=10>"
           'g_iDbRoomFound = True
           'Set docLogEntry = g_vwLogDbUsage.GetLastDocument
       'End If
       'Set docLogEntry = g_vwLogDbUsage.GetNextDocument(docLogEntry)
   'Wend
   End If
```

Now the agent brings all of the preparation into effect. It sets a number of global variables which will be returned to the initialize event of the main agent:

```
   g_sDbSizeText = sDbTextPre & "&nbsp;Size: " & sDbSizeTextMid & sDbTextPost
   g_sDbMonthUsesText = sDbTextPre & "&nbsp;Uses this month: " & sDbMonthUsesTextMid &
sDbTextPost
   g_sDbAgeText = sDbTextPre & "&nbsp;Age of database: " & sDbAgeTextMid & sDbTextPost
```

The fuction ends:

```
End Function
```

7.6.3.7 fnFormatURL Function

This function runs a very simple search and replace, finding backslashes and replacing them with forward slashes.

For example, you could call the function with the following line:

```
fnFormatURL( "one\two\three" )
```

And receive the following result:

```
"one/two/three"
```

This function is used to create valid URL strings:

```
Function fnFormatURL( sURL As String ) As String
   'format like this: fnFormatURL( "one\two\three" )
   Dim sFSlash As String
   Dim sBSlash As String
   Dim sDbLabel As String
   Dim iSlashPos As Integer
   sFSlash$ = Chr(47)
   sBSlash$ = Chr(92)
   iSlashPos = Instr(1, sURL, sBSlash$)
   While iSlashPos <> 0
      Mid$( sURL, iSlashPos, 1) = sFSlash$
      iSlashPos = Instr(1, sURL, sBSlash$)
   Wend 'sDbPath is now a string formatted for the web - with forward slashes
   fnFormatURL = sURL
End Function
```

7.6.3.8 Sub subGetPlace

This sub looks for Places in a Notes DatabaseDirectory. This technique uses the directory structure to define a group of databases, and by looping through this collection, we can find databases.

This sub builds HTML to display a Place and, each time it finds one, branch off and find the rooms that belong to it. This sub is called by the Initialize sub to find all Places. When a Place is found, this sub calls another sub called subGetRoom.

subGetRoom does a similar thing to this sub, that is, build HTML. However, the way it is called and the formatting of the HTML from subGetPlace and subGetRoom are different.

First, the sub declares a number of variables:

```
Sub subGetPlace
   Dim dirPlace As NotesDbDirectory
   Dim ndbPlace As NotesDatabase
   Dim sPlaceTemp As String
   Dim sPlaceName As String
   Dim sNdbPlaceFilepath As String
   Dim sHTMLPlaceIn As String
   Dim sHTMLPlaceOut As String
   Dim sHTMLAfterPlaceTitle As String
   Dim iFolderNameLength As Integer
```

Now that we have the variables declared, we can start building the HTML that will be returned by the sub:

```
   sHTMLPlaceIn  = |<tr height=5>
      <td width=40 height=5><img src=../ecblank.gif width=40 height=5></td>
      <td width=200><img src=../ecblank.gif width=200 height=5></td>
      <td width=100%><img src=../ecblank.gif width=200 height=5></td>
   </tr>
   <tr><td width=40>&nbsp;</td>
      <td class=m-place-text><img src=../icnPlace.gif align=absmiddle width=17
height=14 border=0>&nbsp;|
   sHTMLPlaceOut = |</td></tr>|
   sHTMLAfterPlaceTitle = |</td><td width=100%>&nbsp;<!-- aftertitle 1 --></td></tr>|
```

Using a while loop and a couple of nested if statements, we can retrieve all the necessary information to find Places:

```
   Set dirPlace = New NotesDbDirectory( g_sServerName )
   Set ndbPlace = dirPlace.GetFirstDatabase( DATABASE )
```

We begin a loop through all databases:

```
   While Not ( ndbPlace Is Nothing )
      sNdbPlaceFilepath = ndbPlace.FilePath
```

We test to see if the path to the database begins with “quickplace”; this indicates that it is in the QuickPlace folder on the serve:.

```
      If Instr(1, Lcase( sNdbPlaceFilepath ), |quickplace\| ) Then
```

We test to see if the database’s file name ends with “Main.nsf”., in other words, it is a Place:

```
         If ndbPlace.FileName = "Main.nsf" Then
```

Now that the agent has found a Place, it can start building HTML to map the Place:

```
            g_sURLTemp = "<a href=/" & fnFormatURL( sNdbPlaceFilepath ) & "
class=m-place-text>"
            sPlaceName = ndbPlace.Title
```

```
                sPlaceTemp = sHTMLPlaceIn & g_sURLTemp & sPlaceName & "</a>" &
sHTMLAfterPlaceTitle
                g_iPlaceCount = g_iPlaceCount + 1
                sNdbPlaceFilepath = ndbPlace.FilePath
                iFolderNameLength = Len( sNdbPlaceFilepath ) - 19
                g_sFolderName = Lcase( Mid( sNdbPlaceFilepath , 11 , iFolderNameLength ) )
& |\|
                Call fnGetDbStats( sNdbPlaceFilepath )
                sPlaceTemp = sPlaceTemp & g_sDbSizeText & g_sDbMonthUsesText &
g_sDbAgeText
                g_sHTML = g_sHTML + sPlaceTemp
                sPlaceTemp = ""
```

The Place has been found, and the HTML has been created to render it in Web browser, so we now go looking for Rooms belonging to it:

```
                'now we have a main, get its rooms
                Call subGetRoom
```

We end the loop's if statements and end the sub:

```
                Set ndbPlace = dirPlace.GetNextDatabase
            End If
        End If
        Set ndbPlace = dirPlace.GetNextDatabase
    Wend
End Sub
```

7.6.3.9 Sub subGetRoom

This sub looks for Rooms in the NotesDatabaseDirectory in the same way that subGetPlace does. However, this sub creates a separate Database Directory because after it is finished, the flow of control must return to subGetPlace, the calling sub.

Another difference between subGetRoom and subGetPlace is that subGetRoom does not call any other subs; it simply returns back to the subGetPlace sub which directly called it.

We do not describe subGetRoom in as much detail as subGetPlace because, apart from the differences previously mentioned, it follows roughly the same structure as the subGetPlace.

First the sub declares a number of variables:

```
Sub subGetRoom
    Dim dirRoom As NotesDbDirectory
    Dim ndbRoom As NotesDatabase
    Dim sRoomTemp As String
    Dim sRoomName As String
    Dim sNdbRoomFilepath As String
    Dim sTempFilepath As String
    Dim sHTMLRoomIn As String
    Dim sHTMLRoomOut As String
    Dim sHTMLAfterRoomTitle As String
```

Now that we have the variables declared, we can start building the HTML that will be returned by the sub:

```
    Set dirRoom = New NotesDbDirectory( g_sServerName )
    Set ndbRoom = dirRoom.GetFirstDatabase( DATABASE )
    sHTMLRoomIn = |<tr height=5>
    <td width=40 height=5><img src=../ecblank.gif width=40 height=5></td>
    <td width=200><img src=../ecblank.gif width=200 height=5></td>
    <td width=100%><img src=../ecblank.gif width=200 height=5></td>
</tr>
<tr><td width=40>&nbsp;</td>
<td class=m-room-text>&nbsp;&nbsp;&nbsp;<img src=../icnRoom.gif align=absmiddle width=17
height=14 border=0>&nbsp;|
    sHTMLRoomOut =  |</td></tr>|
    sHTMLAfterRoomTitle = |</td><td width=100%>&nbsp;<!-- aftertitle 1 --></td></tr>|
```

We begin a loop though all databases, building the HTML which will display in the final HTML document:

```
    While Not ( ndbRoom Is Nothing )
        sNdbRoomFilepath = ndbRoom.FilePath
        sTempFilepath = |quickplace| & g_sFolderName & |pagelib|
        If Instr(1, Lcase( sNdbRoomFilepath ), sTempFilepath) Then
            sRoomName = ndbRoom.Title
            sRoomTemp = sNdbRoomFilepath
            g_sURLTemp = "<a href=/" & fnFormatURL( sRoomTemp ) & " class=m-place-text>"
            sRoomTemp = sHTMLRoomIn & g_sURLTemp & sRoomName & "</a>" &
sHTMLAfterRoomTitle
            'If Instr( 1 , sNdbRoomFilepath , "PageLib") Then Print( sNdbRoomFilepath & "
- Roomfilepath, should be in next comparison")
            Call fnGetDbStats( sNdbRoomFilepath )
            sRoomTemp = sRoomTemp & g_sDbSizeText & g_sDbMonthUsesText & g_sDbAgeText
            g_sHTML = g_sHTML + sRoomTemp
            sRoomTemp = ""
            g_iRoomCount = g_iRoomCount + 1
        End If
        Set ndbRoom = dirRoom.GetNextDatabase
    Wend
End Sub
```

We haven’t worked very much inside Places in this example. We never went further than the level of rooms which match individual Domino databases. However, we have seen how we can add useful funcitionality to our QuickPlace server through Domino agent programming.

In the final sections, we discuss some PlaceBot debugging and server performance considerations.

7.6.4 Debugging a PlaceBot

As you define and test a PlaceBot, you may need to make some adjustments in order to get the PlaceBot to run successfully. The following are some debugging tips for problems you might encounter.

7.6.4.1 Compiling a LotusScript PlaceBot
When you run a LotusScript PlaceBot, the QuickPlace server compiles the code before executing it. Thus, the first error you might encounter would be a LotusScript compilation error or warning. The errors and warnings can be difficult to understand. For detailed information, refer to the *Lotus LotusScript Release 3.1 Language Reference*, which is available as a downloadable file from:

```
http://www.notes.net/notesua.nsf
```

If you get a compilation error, you can click the Try Again link to re-import the agent file after the compilation errors have been fixed.

7.6.4.2 Compiling a Java PlaceBot
If you import Java source files (.java), you may get compilation errors when the PlaceBot is submitted to the QuickPlace server. For details on Java error messages, refer to Java documentation.

If you get a compilation error, you can click the Back button of your browser to return to your QuickPlace.

7.6.5 Performance considerations with PlaceBots
These considerations are the same as with Domino agents.

7.6.5.1 Scheduled agents
Stay away from short intervals, as this will drain the processing power of the server.

7.6.5.2 Form PlaceBots
Use this conservatively, as compiled code must be loaded or verified each time.

7.6.6 Disabling all PlaceBots on a server
The way to disable PlaceBots on a server is different, depending on whether you are running QuickPlace on a standalone server, or if you are running it as an overlay (on a Domino server).

7.6.6.1 Disabling all PlaceBots on a standalone server
If you disable Placebots, no PlaceBots will run. The UI for creating and editing PlaceBots will also be disabled.

Disabling PlaceBots on a standalone installation is done in the server administrator's Place of a QuickPlace server:

To disable PlaceBots on an entire QuickPlace server take the following steps:

1. Enter the administrator's Place. This is accessed by signing in as administrator on the QuickPlace Welcome page.
2. Click **Server Settings**.
3. Go into **Other Options** and click **Edit Options.**
4. Deselect the **Enable PlaceBots** check box.
5. Click the **Next** button.

7.6.6.2 Disabling all PlaceBots on an overlay server

To disable PlaceBots on a QuickPlace server running as an overlay on a Lotus Domino server, do the following:

1. Complete Steps 1 to 4 in 7.6.6.1, "Disabling all PlaceBots on a standalone server" on page 214.

   This will only enable/disable QuickPlace *Form* PlaceBots. To enable/disable QuickPlace *Scheduled* PlaceBots, you need to load/unload the agent manager process using the Lotus Domino server console.

   Note: The loading/unloading of the agent manager process will affect the operation of *all* Lotus Domino scheduled agents.

2. Open the notes.ini file for the server (we found our INI file in the directory D:\Lotus\Domino\).

3. Make a backup of the Notes.ini file.

4. Find the ServerTasks line. Ours looked like this:

```
ServerTasks=Replica,Update,Stats,AMgr,Adminp,Sched,Event,HTTP
```

5. Remove the task AMgr. and save the Notes.ini file.

7.7 Summary

In this chapter we looked at PlaceBots, which essentially are the same as agents on a Domino server.

We also provided examples that provide useful functions for administrators and users, while at the same time allowing programmers to learn about QuickPlace internals.

Chapter 8. QuickPlace Object Model

In order to customize QuickPlace, you need to know some of the QuickPlace internals. In this chapter we describe the QuickPlace Object Model and discuss how you can access it programmatically. We also include other general programming tips and a brief description of the tools we used when writing this book.

You can read this chapter and understand the general concept of the QuickPlace Object Model without any particular prerequisites. Because QuickPlace is built on top of Lotus Domino, knowledge of the Domino Object Model will help, but is not required. We will also discuss techniques that require knowledge of JavaScript. Some of these techniques are related to customization topics, such as customized Themes, customized HTML Forms, and so on.

8.1 Why understanding the QuickPlace Object Model is important

The purpose of describing the QuickPlace Database Object Model is to give you an understanding of the underlying structure of QuickPlace.

In the context of an application, Objects refer to the building blocks forming the structure, logic, and content of that application. For example, in the JavaScript Object Model, the browser Window is an Object, and an InputField is an Object contained in a Document Object. In QuickPlace, an example of an Object is a Place, or a PlaceBot.

Most changes to the user interface are made using Web authoring tools, and by using the online tools QuickPlace makes available. For example, implementing a Theme changes the layout of QuickPlace components. When a Theme is implemented, we are *rearranging* the components, not changing them. To make changes at this level, it is not necessary to understand the QuickPlace Object Model. For example, if you create a new Theme, you will use HTML skills and perhaps some DHTML or JavaScript.

However, sometimes you may want to edit or replace images that you see in default QuickPlace pages, for example, the green “Go” image that appears next to the Simple Search field. Although this image appears in pages positioned via HTML layout files (refer to Chapter 4.3, “The QuickPlace Theme layout architecture” on page 75), it cannot be changed by changing your HTML layout files.

In order to change this image, you first need to find the image on the QuickPlace server and then understand what effects the changes you make will have on other QuickPlaces. Any changes you make to the Go.gif will be server-wide. In other words, *that image will be changed in all* QuickPlaces. Therefore, finding the image and understanding the ramifications of changing it require an understanding of the QuickPlace Object Model.

Additionally, if you are making changes to the underlying structure and logic of a QuickPlace, you also need to have an understanding of the QuickPlace Object Model. This includes understanding that object's dependencies. For example, if you create a PlaceBot, you will be manipulating QuickPlace objects. These objects do not work in isolation, and it is important to understand their function within the entire QuickPlace application.

Another reason why understanding the QuickPlace Object Model is important is so that you can take advantage of the existing Domino design structure. Although QuickPlace's Object Model is a model independent of Domino, it is implemented using core Domino technology. QuickPlace utilizes Domino's strengths and extends them to make a robust platform specifically for creating powerful Web applications. Understanding the QuickPlace Object Model means that you can harness Domino features with a minimum of effort, and will not have to "re-invent the wheel".

Virtually every part of the QuickPlace Object Model can be modified. However, our experience has been that in any one project, you will typically only need to manipulate a few objects, at most. QuickPlace has some powerful built-in tools. Knowing how to access those tools maximizes the existing potential with the smallest amount of effort. Reuse, don't reinvent.

8.2 QuickPlace Object Model and the Domino Object Model

Developers familiar with the Domino Object Model will be able to leverage their existing skills when developing on the QuickPlace platform. *PlaceBots*, for example, are actually Domino agents, and it is possible to create and test them on Domino databases. Within the QuickPlace Object Model, however, there are some divergences from the Domino Object Model. If these differences are not understood, they can lead to confusion among developers familiar with Domino.

An example of where the QuickPlace Object Model can be confusing for Domino developers is with *QuickPlace Forms* because they are not the same as *Domino Forms*. QuickPlace Forms more closely resemble Domino documents, because they are created using a Domino Form, and contain a

Domino Text Field with a value of *h_Form*. The value of h_Form tells QuickPlace that this Domino Document should be rendered in a browser as a QuickPlace Form.

This structure provides a lot of flexibility for Web applications with less complexity than if Domino Forms were used. For example, in a default QuickPlace, a user can create a new QuickPlace Form. The user chooses which fields they want in the form, in what order they should appear, and what text and or graphics should appear near them. To create this sort of instant structure on the Web using Domino Forms would be very complex indeed.

QuickPlace has extended this concept of being able to use HTML to define Forms by enabling the creation of custom QuickPlace Forms using imported HTML. These Forms not only make use of Web authoring technologies such as JavaScript, but also have the back-end support of Domino. This back-end logic is implemented via tools such as Placebots (Domino Agents). This means that Forms have the ability to not only to define the look and feel of visible parts of an application, but also have the potential to initiate workflow, and have many other powerful automated features.

Domino Forms are designed to support Web browser clients and the Notes client. In order to do this, Domino Forms must have certain features not required by a Web client. QuickPlace Forms, on the other hand, have been optimized by stripping away many of the Notes features not required when used on the Web.

Yet another advantage of this structure is the fact that it enables the use of Web authoring tools to extend the objects. Returning to our example of QuickPlace forms, it is possible to modify forms using XML, JavaScript and HTML and any other Web tools. Knowledge of JavaScript and HTML are more common than Domino Designer skills, thus making the QuickPlace an easily accessible platform for Web programmers and designers.

The divergence of the QuickPlace Object Model from the Domino Object Model discussed here in relation to QuickPlace Forms applies to some other objects in QuickPlace. However, it is important to note that in general, when the QuickPlace Object Model diverges from the Domino Object Model, other Domino Objects are used. Once again using Forms as an example, a QuickPlace Form maps to a Domino Document. Once we understand that, it is possible to leverage our Domino development skills and all the functionality of Domino Documents.

Some parts of the QuickPlace Object Model implement Domino/Notes functionality in slightly different ways to a standard Domino application. For example, QuickPlace uses Domino's security and authentication model as a basis for its management of access to QuickPlaces. However, instead of primarily utilizing the Domino Directory, QuickPlace also uses a Contacts1.nsf database for each QuickPlace. To understand this, it is helpful if you are familiar with the Notes security model, in particular with basic access control list (ACL) settings, and the use of Reader and Author fields.

8.2.0.1 The QuickPlace Object Model figure

The QuickPlace Object Model shown in Figure 85 on page 221 describes QuickPlace objects, how they relate to the Domino Object Model, the directory structure and how Objects relate to each other within the hierarchy.

The purpose of illustrating this model is to give a visual representation of the containment and association between objects.

The highest level of the model is the QuickPlace Server, and within the server are all the QuickPlaces, as well as the resources they access to finally render Web applications.

Note that Figure 85 on page 221 only illustrates the Model focusing on QuickPlaces. To see the extended QuickPlace Object Model, refer to Figure 140 on page 401. The extended Model shows the resource databases used for instantiation of QuickPlaces and the management of the QuickPlace server.

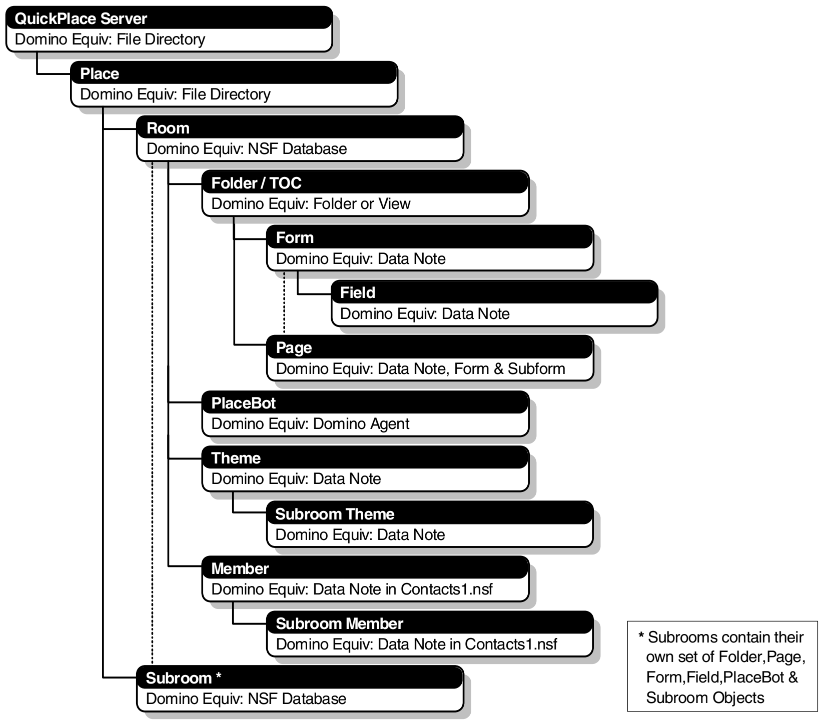

QuickPlace Server
Domino Equiv: File Directory
Place
Domino Equiv: File Directory
Room
Domino Equiv: NSF Database
Folder / TOC
Domino Equiv: Folder or View
Form
Domino Equiv: Data Note
Field
Domino Equiv: Data Note
Page
Domino Equiv: Data Note, Form & Subform
PlaceBot
Domino Equiv: Domino Agent
Theme
Domino Equiv: Data Note
Subroom Theme
Domino Equiv: Data Note
Member
Domino Equiv: Data Note in Contacts1.nsf
Subroom Member
Domino Equiv: Data Note in Contacts1.nsf
Subroom *
Domino Equiv: NSF Database
* Subrooms contain their own set of Folder,Page, Form,Field,PlaceBot & Subroom Objects

Figure 85. Map of the QuickPlace Object Model

The following section describes the Objects listed in the previous outline in more detail. Written from a programmer’s perspective, it focuses on each of the Objects to give practical information on how you can access them in your applications.

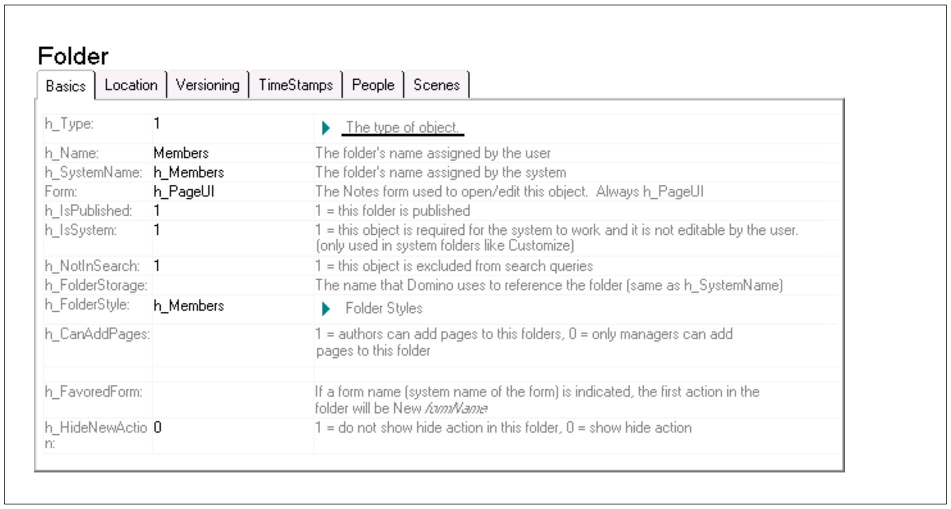
8.2.1 Why use design notes

QuickPlace uses design notes for many Objects in QuickPlace so that Objects in the Place can be organized more easily. In the following text we use the Folder object as an example. For a list of Objects that are accessed via design notes, see the Object Model map.

As an example of how design notes are implemented, we just need to look at the Table Of Contents in a Place. The Table Of Contents is a list of pages, folders and tools such as the Customize Area. Domino Folders cannot be listed in Domino views so they must be listed using a link document, or Note.

To see the different data notes in a QuickPlace, open the Domino database for the room you are interested in (*Main.nsf* or *PlaceLibrary...nsf*), using the Notes Client. Open the view named **QDK**. Here you can see the type, system name, title and last modification date for all the documents. When you open a document in the QDK view (it does not work with any of the other views), it is displayed in Notes using a Form that allows you to see the most important fields for that document.

For example, Figure 86 shows a Folder Note opened in a Notes client using the QDK folder Form *h_Folder*. This data note describes the folder and tells QuickPlace how to link to the folder.



h_Type:	1	▶ The type of object.
h_Name:	Members	The folder's name assigned by the user
h_SystemName:	h_Members	The folder's name assigned by the system
Form:	h_PageUI	The Notes form used to open/edit this object. Always h_PageUI
h_IsPublished:	1	1 = this folder is published
h_IsSystem:	1	1 = this object is required for the system to work and it is not editable by the user. (only used in system folders like Customize)
h_NotInSearch:	1	1 = this object is excluded from search queries
h_FolderStorage:		The name that Domino uses to reference the folder (same as h_SystemName)
h_FolderStyle:	h_Members	▶ Folder Styles
h_CanAddPages:		1 = authors can add pages to this folders, 0 = only managers can add pages to this folder
h_FavoredForm:		If a form name (system name of the form) is indicated, the first action in the folder will be New *formName*
h_HideNewAction:	0	1 = do not show hide action in this folder, 0 = show hide action

Figure 86. Design Note for a Folder Opened in a QDK Form

As we describe the different objects in this chapter, you can open the matching objects in your QuickPlace and see their values.

We will now discuss the objects in the QuickPlace Object Model.

8.2.2 QuickPlace server

A *QuickPlace server* is a file directory containing all Places and Resources. The Domino equivalent is a file directory named quickplace.

This is the main folder for a QuickPlace server. If the QPserver is running as a standalone, this folder will be in the QuickPlace Data folder, for example D:\QuickPlace\Data\quickplace. If the QPserver is running on top of a Domino server, the folder will be the Domino Data folder, for example D:\Lotus\Domino\Data\quickplace.

The QuickPlace server in PlaceBots

To locate the QuickPlace server and get access to all its databases, you can use the LotusScript method GetDbServer and test that the Path to databases starts with the term: quickplace.

```
Dim dirPlace As NotesDbDirectory
Dim ndbPlace As NotesDatabase
Dim sNdbPlaceFilepath As String
Set dirPlace = New NotesDbDirectory( "MyServer/Server")
Set ndbPlace = dirPlace.GetFirstDatabase( DATABASE )

While Not ( ndbPlace Is Nothing )
    sNdbPlaceFilepath = ndbPlace.FilePath
    If Instr(1, Lcase( sNdbPlaceFilepath ), |quickplace\| ) Then
        If ndbPlace.FileName = "Main.nsf" Then
            '//you now have a QuickPlace
```

8.2.3 The Place object

The *Place Object* is a directory in the “QuickPlace directory” grouping resources for a Place. The Domino equivalent is a file directory bearing the name of the QuickPlace.

The Place Object is a directory that brings together a Place for organizational purposes. It also identifies the NSFs as belonging to the Place by bearing the name of the QuickPlace.

The Place Object is often confused with the Main Room of a Place. The main Room in a QuickPlace is a database called Main.nsf. The Place Object groups and identifies the Main.nsf resources for the Place, and any Subrooms in the Place.

The Place Object contains several files. There is a Main.nsf, a Contacts1.nsf, and a Search.nsf file. If your QuickPlace has a Subroom, there will also be an NSF file whose name starts with PageLibrary. Each of these page library files is a Room.

The Place Object in PlaceBots

The Place Object (directory) contains the databases which form a Place. When you are writing PlaceBots, you can search for this directory by using the name of the QuickPlace. In this directory, you can find all databases that belong to that Place.

This file directory's name is the name of the QuickPlace. For example, if your QuickPlace is called “Millennia”, this directory has the following path within the QuickPlace Server: `\millennia`

To find the Place Object for the Millennia Place in LotusScript, you can use a script like the following:

```
Dim ndbPlace As NotesDatabase
Set dirPlace = New NotesDbDirectory( g_sServerName )
Set ndbPlace = dirPlace.GetFirstDatabase( DATABASE )
sNdbPlaceFilepath = ndbPlace.FilePath
If Instr(1, Lcase( sNdbPlaceFilepath ), |quickplace\millennia| ) Then
'//you now found the Place
```

Note: We make the comparison of the file paths in lowercase to make it more robust on different server platforms.

8.2.4 The Room Object

The *Room Object* is the main container for a Place, containing a collection of pages and tools. The Domino equivalent is the NSF Database.

The Room is the main container for a QuickPlace's content. For example, when using the Millennia Place, most of what we see is contained in the Room Object. The Room Object is always called Main.nsf, and holds Folders and Pages for the QuickPlace, as well as managing links to any Subrooms in the Place Object.

A Room breaks down a Place into smaller areas to help define structure.

The Room Object uses elements held in other databases. For example, many of the standard images QuickPlace displays are in the Resources Object. Each Room has its own security and authentication, and the information required to do this is contained in databases such as Contacts1.nsf.

The fact that each Room has its own security and authentications is important to site managers, because it allows separate user groups. It also means that Subrooms can be created for separate projects, forming a separate shared space. The Room object then forms a common entry point where shared resources can be stored.

The Room Object in PlaceBots

To locate a Room, you must look in the main QuickPlace Server directory, then look into the Room Object (a directory bearing the name of the QuickPlace), then look for a database called *Main.nsf*.

Note: Are you wondering why we only mention Main.nsf and not the PageLibrary..nsf databases? This is because these databases represent *Subrooms* in the object model. See 8.2.14, "The Subroom Object" on page 239 for more information on Subrooms.

Returning to our previous LotusScript example of locating a Place, we can extend the match string from “quickplace\millennia” to “quickplace\millennia\main.nsf” to find the Room Object:

```
Set dirPlace = New NotesDbDirectory( g_sServerName )
Set ndbPlace = dirPlace.GetFirstDatabase( DATABASE )
sNdbPlaceFilepath = ndbPlace.FilePath
If Instr(1, Lcase( sNdbPlaceFilepath ), |quickplace\millennia\main.nsf| ) Then
'//you now found the Room
```

You access elements contained in a Room via the views and folders in the Room. For example, if you want to find the elements visible in the Table Of Contents, you can access the h_Toc view.

For more information on using views to locate database elements see 8.2.5, “The Folder Object” on page 226.

The Room Object in HTML

The Room Object is visible in URLs as the Main.nsf. To access the Room Object most easily, you can use a relative path from the current Object if it is in the same Place. For example, you are creating a URL link from a Subroom to a Room, you can start the URL like this:

```
<a href="../../Main.nsf/
```

Note: The “dot dot slash dot dot slash” syntax is a part of the URL, not an abbreviation for this example.

Using this relative URL makes your URL more robust. In other words, this URL can be used to find the (Main.nsf) Room for any Place.

Room fields

The fields shown in Table 30 are used to define Rooms.

Table 30. Fields used to define Rooms

Field name	Description
h_HaikuName	The name of this Place.
h_AreaType	The name of the template used to create this room.
h_AreaParent	The name of the parent database.
h_ShowSecurity	If h_SetSecurity = 1, the QuickPlace server sets h_ShowSecurity to 1.
h_SetCalendar	Determines if the Calendar will be visible in a Room. If the field has the value of “1”, a link to the Calendar will be displayed in the sidebar.

Field name	Description
h_SetSecurity	This field works in conjunction with the h_ShowSecurity field. It is only valid for Readers and Authors, because Managers must always be able to edit security of a Room. If the field is set to “1”, a link to the Security page will be displayed in the sidebar for Readers and Authors (if they select Security in this case, they will see only their own information).
h_MailDb	The name of the database that receives email addressed to this Place.
h_LastAttachmentDirectory	Last directory used when getting attachments. This field enables users to quickly upload attachments. For example, you might notice that each time you upload a Layout file, QuickPlace knows where to go looking for the file. This path information is sourced from this field.
h_DirtyAesthetics	Number which indicates which items should be checked (once you have tweaked a part of the aesthetics, the check mark indicates that you already changed that part).
h_AreaHasAesthetics	This field indicates if a Room has its own aesthetic settings enabled. If the field value is “1” ,the Room has had the aesthetics tweaked.

8.2.5 The Folder Object

The *Folder Object* is used for indexing content, grouping related pages, and dividing a Room into sections, without imposing new security. The Domino equivalent is the Notes Folder or View and Note.

Folders can be used in three ways. First of all, they provide a logical grouping of related documents for users. This makes it easier for users to find documents, and allows people with a shared interest to work in an area of a QuickPlace.

The second way of using folders is in the user interface, or User Folders. Within User Folders there are seven different types:

1. Standard List
2. Headline
3. Slide Show
4. Response List
5. Ordered List
6. Table Of Contents
7. Index

Folder types 1 to 5 are all available as styles for new, custom folders.

From the a site manager's perspective, a Folder allows a QuickPlace to be divided into areas for separate groups of people, without having to be concerned about Access Control (which *would* be necessary if a Subroom were used).

The third way that Folders are useful is for developers; they allow the developer to locate elements in a QuickPlace. To a developer, Folders are indexes that allow lookups, therefore giving programmatic access to elements.

Hint: When any page renders in a browser, the time it takes to render is directly dependent on the amount of information to be downloaded. The amount of information required to render a Folder is less than for a Page. When Pages appear in Edit mode, there is yet more information required to render it. Therefore, to get the quickest load time for a QuickPlace, use a Folder as the first page the user sees when they arrive at the Place. Once users have visited a Folder, a subset of the resources used to render a Page will already have been downloaded.

The Folders used by developers are slightly different from the Folders users would use. Note that the h_Index lists the published pages in the Place and appears as the standard index of a Place, and the h_Toc is the table of contents list.

Table 31 lists some of the folders commonly used in lookups by developers.

Table 31. Views commonly used by developers to reference Objects

View name	Description
h_Index	Provides a list of all published Pages in a Room, listed by h_UNID, the unique identifier for a Page.
Haiku	Lists all published items in a Room, which not only includes Pages but all of the Objects in a Place (for example, Pages, PlaceBots, Fields, Themes and Forms).
h_QDK	Every Design Note in a Place. The h_QDK view contains a Form formula to open different documents using different Forms. For example: If the field h_Type is "0", then use the form named h_Page. The result of this Form formula is that the QDK view allows developers to inspect the properties of some Design Notes. The supported types are: h_Page, h_Folder, h_Room, h_SubRoom, h_Error and h_RoomType.

View name	Description
h_TOC	List of all items displayed in the Table Of Contents. Items must have the h_IsInTOC field with a value of "1" and be published with no replication-save conflict.
(All)	Every item in the Room. Sorted by the h_Name field: the readable name of the item (for example “Welcome”, representing the default Welcome page).

The Folder Object in PlaceBots
Internally, default QuickPlace Folders have readable titles. For example, the response folder Discussion has the internal name of h_Discussion in the h_SysName field.

We also created a new response list style folder called SchwatzRaum (“chat room” in German). The internal name of the SchwatzRaum Folder is: h_F49791727035ACD1C12569510063087C (which does not mean anything in German). This unique identifier can be used in PlaceBots to locate the Folder.

Hint: You can do a lookup in the h_Folders view of a QuickPlace to find the readable name of the folder. A better solution is to retrieve the name of the field by accessing the value in the h_SysName field.

The Table Of Ccontents and the Index are special user Folders. Only one TOC and one h_Index exists per Room or Subroom. They exist from the moment the Place or Room is instantiated, and you cannot change them.

Folder fields
The following fields are used to define data notes that render as Folders. Folders exist in a visible form within a QuickPlace. In other words, they can be viewed by opening the NSF file in the Notes Client or Domino Designer. In conjunction with this view, a data note exists, providing information about that

Domino View or Folder. Table 32 shows the fields that are contained in the data note and which provide information about the Domino View or Folder.

Table 32. Fields used to define Folders

Field name	Description
h_FolderStyle	When creating a new folder, you are given the choice to create a new folder based on a number of templates. This field denotes which type of folder has been created. “1” = Standard List “3” = Headline “4” = Slide Show “5” = Response List “7” = Ordered List
h_FolderStorage	The “internal” name of the folder; in other words, the name by which it is known to the system. The value of this field is used in documents to tell QuickPlace in which folder it should be used.
h_CanAddPages	When creating a new folder, you are presented with options to the question “Who can add pages to this folder?”. If you choose Only managers, the value of “0” is written to this field. The default is ““ which means all authors can add pages to this folder.

8.2.6 The Form Object

The *Form Object* is the document used to create new QuickPlace content. The Domino equivalent is a data note of the type h_Form.

The Form Object is a resource used to create, manage and display content, therefore it defines the schema of the application. Forms contain fields to hold data, therefore creating and displaying content.

Forms can also contain scripts within them to provide logic within the Page. For example, a Form can contain form validation to make sure that a field contains only numbers.

Forms can also initiate processes outside the Page. This is done by creating a PlaceBot and associating a PlaceBot with a Form. PlaceBots are not contained by the Form, but there is a association between them.

Forms are created with the Domino Form h_PageUI with the field h_Type set to h_Form.

New Forms with custom structure and logic can be created by Room Managers. For more information on this, see Chapter 6, “Forms in QuickPlace” on page 135.

Form fields

Table 33 shows the fields used to define the structure of a Form.

Table 33. Fields used to define Forms

Field name	Description
h_FormDescription	The content of this field appears as the description of the form appearing in the “New” page.
h_WorkflowType	h_ApprovalCycle Allows 1 to 4 approvers and some other options. This is normally set to h_Standard.
h_EditorInChief	Allows 1 approver and fewer options.
h_MultipleEditors	By setting this field, you can allow all members of QP to edit pages created with this form.
h_Standard	None of the above.
h_SetPageComponent sView	Should = h_FieldDefinitions.
h_IsUserDefined	h_True means this is a custom form.
h_PublishInFolder	UNID of the folder + "I" + h_FolderStorage name of the folder.

8.2.7 The Field Object

The *Field Object* is used to construct (HTML formatted) input fields in Forms. The Domino equivalent is a data note of type h_Field.

Field data notes are used to construct (HTML formatted) input fields in Forms. Fields are constructed from the Domino Form h_PageUI with the field h_Type set to h_Field.

The Field Object in PlaceBots

Fields are not often manipulated when customizing a QuickPlace. In order to understand why this is, you first need to separate *fields* (container) from *field content* (the stuff inside the fields). A QuickPlace Field Object defines the structure of the container, not the content.

The values contained in a Page are contained by the *Page*, not the *Fields*.

For more information on manipulating field values programatically, refer to 8.2.8, “The Page Object” on page 233.

The h_FieldType attribute of a Field determines what sort of field it is. This is important because it determines what the field will do when it is rendered in a browser. For example, a Field of type h_DateControl will provide the user with a date picker widget.

Fields used to define QuickPlace Fields

The Domino Fields are used to define the attributes a QuickPlace Field has; see Table 34. For Domino programmers, this may seem a little strange. To understand Fields in QuickPlace, one must first of all know that QuickPlace Fields are drawn to the screen as HTML, they are not created by a Domino Field in a Domino Form. For more information on this subject, refer to 8.2, “QuickPlace Object Model and the Domino Object Model” on page 218.

Table 34. Fields used to define QuickPlace Fields

Field name	Description
h_Name	“Import” and is related to the h_SystemName field, which often has a similar value such as h_Import.
h_FieldLabel	Instructional information that might be useful for someone editing this field. Similar to the Static h_FieldType. It contains information to help the user, but is only displayed in edit mode." For example: <script> (h_CurrentSkinType == 'h_Edit') ?"": C(self, 'Note: Clicking on the title of this page in its folder or in the sidebar will open the page that it points to. To edit the page again later, click its title in the Index.'); </script>"
h_ContainerUNID	The UNID of the Form which contains this field. QuickPlace uses a Design Note to create Forms, each of these having an internal name. The h_ContainerUNID contains the internal name of one of these QuickPlace Forms.

Field name	Description
h_FieldType	There are many different types of Fields. There are too many to describe in this section. The following types are listed as examples to help understand how Fields work in general. "h_Attachments"= Enables the attaching of files. "h_CalendarControl"= Includes date and time controls and a duration field. "h_DateControl" = Date field with date picker widget. "h_DateTime" = Contains Date and Time information. "h_DocAuthor" = Contains a Domino hierarchical name of the original Author of the Document. "h_DocCreated" = Creation date of the page. "h_DocModified" = Modified date of the page. "h_DocSize" = Size of the page. "h_NamePopup" = Select listing members of the QuickPlace. "h_RichText" = Rich text field. Allowing editing via the rich text editor applet. "h_Serial" = A unique number to identify the document. "h_Static" = Static text, used to provide information about the accompanying field. May also include link to an image. "h_Subject" = The Documents subject. "h_TaskControl" = Used in the Task form to insert the task control tool. "h_TextInput" = Simple text equating to the “<input>” field in HTML. "h_TextPopup" = Text select list, equating to the “<select><option>” in HTML. "h_TimeControl" = Select lists for hours, minutes, AM/PM. "h_CalendarControl" = Field containing control tool used in the calendar field. "h_CreateMSExcel" = Field enabling the upload of Excel documents. "h_CreateMSPowerPoint" = Field enabling the upload of PowerPoint documents. "h_CreateMSWord" = Field enabling the upload of Word documents. "h_Import" = Field enabling the upload of imported documents such as HTML. "h_MultipleImport" = Field enabling the upload of multiple documents, such as a series of HTML documents. "h_NotifyIndicator" = Field indicating if members should be notified of the creation of content or their inclusion in the Contacts1.nsf.

Field name	Description
h_NotInSearch	Having the value of "1" will exclude the field from being included in the full text search. This allows functional content in fields such as JavaScript or static text to evade returning a hit during searching.
h_Position	Indicates the fields position of appearance in a form. Typically numbers such as 100 are used.
h_FieldFormat	"h_FieldFormat" Indicates formatting options, "h_All" "h_BannerOptional" "h_BannerRequired"
h_BannerRequired	Always display subject as a banner at top of page.
h_BannerOptional	Allow user to choose banner or not.
h_NoBanner	Do not display the subject on the page.
h_FieldIsRequired	1 = The field is required and the user will be prompted if they do not fill it out.

8.2.8 The Page Object

The *Page Object* is a basic building block for content. The Domino equivalent is the data note, form and subform.

Pages form the basic units of content, relying on the structure of QuickPlace to create, manage and render them in a Web browser.

Developers familiar with the Domino Object Model will know that it differentiates structure and content cleanly. Domino structural elements such as Forms Views and so on provide structure, whereas Domino documents provide pure data content.

In the Domino environment, the division between structure and content becomes blurred. This is because when the data in a document is being represented in a Web browser, it is possible to use the data to format itself using HTML. The data is able to start defining structure by creating HTML links, tables, references to images and so on. In the QuickPlace Object Model, the same is true.

Pages can be created in a number of ways. This creation process is dealt with in the *QuickPlace User Guide*. This section deals with understanding and manipulating Page content.

Page fields

The fields listed in Table 35 are used to define Page documents in QuickPlaces.

Table 35. Fields used to define QuickPlace Pages

Field name	Description
h_Form	The QuickPlace Form used to create this Page. This should not be confused with the Domino "Form" field, which denotes which form Domino links the file to. The Domino "Form" field will contain "h_PageUI" for virtually all objects in a QuickPlace.
h_PageType	This field is set to null when the document is a visible document. Only when the object is in design mode do the other values appear: "h_Response" means the document is a response to a topic document. This value is only valid in response folders. "h_Revision" means that the document is being revised, and is not available for public access. "h_Mail" means the document is a mail document, being either sent or received by QuickPlace.
h_Originator	The creator of this page. This field contains a full hierarchical name, for example: "CN=David Wyss/OU=QuickPlaceName/OU=QP/O=ServerName". All users have the second OU part of the name set to QP. This is done so that when QuickPlace is used on an overlay server (QuickPlace and Domino together), QuickPlace can avoid conflicts between Domino registered users and QuickPlace users.
h_NameIsBanner	Denotes if the page's name should be displayed as a banner. If it is to be displayed as a banner, this field contains the value "1". Setting this field is done when the user clicks the "Show the title, author and date on page?" checkbox.

Page object in LotusScript and JavaScript

Developers wanting to customize Pages will generally want to manipulate the Page's field values.

Note: Fields existing in a Page are generally rendered to the HTML document in the background as JavaScript variables. They are then visibly rendered via document.write() functions. For a developer, the significance of this is if a field exists, you can access it in the browser via a variable with the same name as the field.

The PageBody Field is an important field for developers because it holds the main content or “body” of the page.

If you are using the PageBody to write out HTML content in a QuickPlace page, this can be done via the JavaScript document.write method. This field can be printed onto the screen via a document.write(PageBody) method called in a QuickPlace document.

The following is an example of using this technique. In a Placebot we write the contents of our document into the PageBody field.

If the PlaceBot has not run, or not run correctly, the PageBody field will be empty. If the document is displayed in a Form where the PageBody JavaScript variable is not declared, you will experience an error.

To avoid an error through an undefined variable, we use the typeof operator. This test assigns a message string to the sPageBodyMessage variable and prints that instead of the PageBody. We recommend that you *do not remove* this error checking.

To customize this message, change the text in quoted on the PageBodyMessage line.

Then we include the following line in the HTML document:

```
<script language=JavaScript>
    if ( typeof( PageBody ) == "undefined" ) {
        var sPageBodyMessage = 'Run the Mapperizer Placebot to see a site map here...';
        document.write( sPageBodyMessage )
    } else{
        document.write( PageBody )
    }
</script>
```

Note: The typeof operator must be in lowercase.

Page Object in HTML

Table 36 lists some of the most commonly referenced JavaScript variables in Pages.

Table 36. Commonly used JavaScript Variables In Pages

Field name	Data type, description
h_Name	String, readable name of the Page.
PageBody	String, content of the Page.
h_SystemName	String, the internal name of a page; for example, 'h_Welcome'

Field name	Data type, description
h_Originator	String, full Notes format name of the document creator; for example: 'CN=Anna Rath/OU=Millennia/OU=QP/O=Server';
h_IsPublished	String, number representing “1” for published or “0“ for not published.
h_LastTimePutAway	String, representing the date and time the Page was last saved '09/03/2000 07:54:08 PM'
Form	String, Domino Form name used to create the Page. Most documents in a QuickPlace are created with the 'h_PageUI' Form. The value that differentiates fields is the h_Type field.
HTTP_COOKIE	String, all cookies available to that Page.
HTTP_HOST	String, name of the server; for example 'millennia.com'
HTTP_REFERER	Page used to send the user to this page.
HTTP_USER_AGENT	String, browser used to access the current Page; for example: 'Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt)'
REMOTE_USER	String, full name of the person reading the Page; for example 'CN=Doug Mudge/OU=Millennia/OU=QP/O=Server';
Server_Name	String, the server name; for example 'www.lotus.com'
h_DocSize	Integer, size of the page; for example: 4705
h_ModifiedDate	String, date and time the page was last saved: for example '09/03/2000 07:54:05 PM';

If you get the user’s name either using the *h_Originator* or the *REMOTE_USER* field, you will get the full hierarchical name like this:

```
CN=Doug Mudge/OU=Millennia/OU=QP/O=Server
```

If you only want to work with the common name part (Doug Mudge), you can use the following JavaScript Function:

```
function fnGetSimpleName(sTxt) {
    iTxtStart = sTxt.indexOf('=');
    iTxtStart++;
    iTxtEnd = sTxt.indexOf('/');
    if(iTxtEnd == -1) iTxtEnd = sTxt.length;
    sTxt = sTxt.substr(iTxtStart,iTxtEnd - iTxtStart);
    return sTxt;
};

return fnGetSimpleName('CN=Doug Mudge/OU=Millennia/OU=QP/O=Server')
```

This JavaScript will return the string *Doug Mudge*.

8.2.9 The PlaceBot Object

The *PlaceBot Object* is a Java or LotusScript Domino Agent, used to create or manipulate QuickPlace Objects automatically. The Domino equivalent is a Domino Agent. For additional information about PlaceBots, see Chapter 7, “PlaceBots” on page 165.

For Java and LotusScript programmers, the PlaceBot is the main way of implementing sophisticated functionality to a QuickPlace. Within the bounds of an HTML document, Web authoring tools such as HTML editors are used. To make links between Objects and manipulate QuickPlace Objects, PlaceBots are used.

8.2.10 The Theme Object

The *Theme Object* is a group of files which define the look and feel of a QuickPlace. The Domino equivalent is a group of data notes. For additional information about Themes see Chapter 4, “Creating Themes” on page 47.

Themes are a mechanism for determining the layout and appearance of a QuickPlace. They also help introduce functionality and, although not their primary function, some content.

There are two types of Themes in QuickPlace: user-defined or “Custom Themes”, and default Themes. To create your own Themes based on the built-in, you can download some of these from the QuickPlace DevZone at:

`http://www.quickplace.com/devzone`

You can also download the Millennia Theme we created for this redbook. See Appendix L, “Additional Web material” on page 425 for instructions on how to get the sample files.

8.2.11 The Subroom Theme Object

The *Subroom Theme Object* is a subset of the Themes in a QuickPlace. The Domino equivalent is a data note.

By default, Subrooms inherit the Theme being used by the (main) Room. Only when the Theme being used in the Subroom has been modified, does it act independently of the Room. To reset any changes, choose the Theme called Reset Theme, with the plain-looking gallery image.

For additional information about Themes see Chapter 4, “Creating Themes” on page 47.

8.2.12 The Member Object

The *Member Object* is a data note listing a user in Contacts1.nsf. The Domino equivalent is a note in contacts1.nsf.

Members are records specifying user access to a Room.

A *member note* contains information about a team member of a QuickPlace. In addition to this data, the member must be listed in the ACL of main.nsf and in a group in names.nsf to pass authentication.

Member fields

The fields listed in Table 37 are used to define Members.

Table 37. Fields used to define Members

Field name	**Description**
h_Password	This member's password. Encrypted with @Password
h_FirstName	This member's first name.
h_LastName	This member's last name.
h_PhoneNumber	This member's phone number.
h_EmailAddress	This member's e-mail address.

Group fields

The fields listed in Table 38 are used to define Groups.

Table 38. Fields used to define Groups

Field name	**Description**
h_Members	The list of members who belong to this group, listed in full hierarchical format.

8.2.13 The Subroom Member Object

The *Subroom Member Object* is a subset of entries in the Main Room of a QuickPlace. The Domino equivalent is a data note in contacts1.nsf.

A Subroom Member has a similar structure to a Room member, but it specifies user access to the SubRoom. These SubRoom Members are a subset of the (main) Room Members list. This means that if you want to grant access to new users, you must first add them as readers (or greater) in the (main) Room.

8.2.14 The Subroom Object

The *Subroom Object* is a container within a QuickPlace, with separate security to the Main Room. The Domino equivalent is the NSF Database.

Subrooms are similar in structure to Rooms. Subrooms are used to create discreet meeting places for a subset of the Members in a Place.

The Subroom object in PlaceBots

To locate a Room, you must look in the main QuickPlace Server directory, then look into the Place Object (a directory bearing the name of the QuickPlace). The Subroom will be named PageLibrary followed by a 16-digit hexadecimal time stamp number, such as "0123456789ABCDEF" then the .nsf suffix.

If we were looking for a Subroom to the Millennia place, we could use the following script:

```
Set dirPlace = New NotesDbDirectory( g_sServerName )
Set ndbPlace = dirPlace.GetFirstDatabase( DATABASE )
sNdbPlaceFilepath = ndbPlace.FilePath
If Instr(1, Lcase( sNdbPlaceFilepath ), |quickplace\millennia\pagelibrary| ) Then
```

We have used the Instr method to look for this database, down to the *PageLibrary* part of the string, because it is difficult to know what the 16-digit number will be.

The Page Object in HTML

To create URLs to reference Subrooms, build the URL in the Main Room using either the h_Area view or the h_Toc view to create the path. This View contains the h_LocDbName field as the first sorted column.

8.2.15 The Resources Object

The *Resources Object* is a database of shared resources. The Domino equivalent is the NSF Database. It is a centralized container for resources required in all QuickPlaces on a server.

Images, Layout files, and Fonts are stored in this database. For example, resources such as the button that appears beside the simple search image "Go.gif" is stored in this database.

The easiest way to find items in this database is by scrolling through the h_SystemNameView. To view elements you can create a dummy form.

Instructions on how to do this are provided in 8.3, “Editing QuickPlace Objects directly with the Notes Client” on page 259.

8.2.16 Common QuickPlace Object fields

In this section we discuss some of the common fields and JavaScript Variables in the h_PageUI Form.

General fields

Table 39 shows the components you can customize for each layout.

Table 39. General fields in the h_PageUI Form

Field name	**Description**
h_Authors	Names of Authors who can edit the document. This is a particularly important field if you are creating PlaceBots which modify the access control to documents.
h_CurrentSkinName	Name of the Theme to be used in the page.
h_CurrentSkinType	Name of the layout file to be used, such as Edit: "h_Edit", or for a custom Theme the ID: "c_E4257D50EE2DD800C12569440019C164"
h_FolderUNID	The system name of the folder the page belongs to; for example: "4695CA1530263B3AC1256946005E965C" - the internal code for a Folder, or ““ when the page only appears in the TOC.
h_Form	The id of the QuickPlace-Form used to create the page; for example: "30DF3123AEFAF358052567080016723D". (The form referred to here is actually a data note and not a Notes Form.)
h_IsInToc	If the page should appear in the TOC, it is set to "1". If it does not appear there, it is set as ““. This is useful if you want to see if the item is visible in the TOC.
h_IsPublished	Set to "1" if the page should be visible to readers.
h_IsSystem	1 = This is a system object.
h_Name	The user visible name of this object.
h_Position	Number used to sort the pages within the TOC. These typically have values such as 10000. This value should be handled as a Long when referenced in LotusScript.

Field name	Description
h_SystemName	The name of this object as known to the system.
h_Type	Describes what sort of note defines. This field is used in all Quickplace Design Notes to tell what sort of document is being referred to. It is what differentiates between the Objects in QuickPlace. "0" = Page “1” = Folder “2” = Room “3” = Subroom “4” = Error Page “5” = RoomType “h_Agent” = PlaceBot “h_Member” = Member “h_HaikuType“ = “h_Group” = Group "h_Form"= Form “h_Field” = Field “h_Skin” = Layout file "h_SkinGroup" = Theme (Skin Group).
h_Name	Name of the page. We used this a few times in the application we wrote. We allow users to create a list of favorites (a type of bookmark created using cookies). When a new bookmark is created, the user is prompted to enter a title for the page. This variable is put into the editable field in the dialog box as the default value.
h_Originator	User name of the creator, such as "CN=User Name/OU=QuickPlaceName/OU=QP/O=ServerName"
h_TextAbstract	The abstract automatically created to summarize the page. This is useful when writing out JavaScript where you want to get a summary of the text content in the document.

Field name	Description
PageBody	The content or “body” of the page. If you are using the JavaScript document.write method to write out HTML content in a QuickPlace element, such as an imported page, Theme and so on, it is normal to do this via the PageBody field. This field can be printed onto the screen via a document.write(PageBody) method called in a QuickPlace document. To do this in a page you have created, use the document.write method to print the contents of this field to the page. When creating content for this field, avoid using the document.write method. If you use this, you will be writing out a write statement. In a Netscape Navigator browser, this will create unexpected results. If errors should occur when the page is being loaded, or document.write methods fail to execute, we suggest that you modify your code and produce dynamic text via some other method. An example of this exists in the site map creating Mapperizer.lss PlaceBot. For more information about the MIME field type, see 8.4, “The MIME Field Type in Domino Designer” on page 261.

System objects are treated specially by the system. They have special meaning, depending on the type of object. The following tables describe fields in various QuickPlace Object types.

Publishing Fields

The fields listed in Table 40 are used to define Publishing status.

Table 40. Fields used to define Publishing

Field name	Description
h_IsPublished	1 = This object is currently published.
h_IsHidden	1 = This object is not shown to the user.
h_SetReadScene	The name of the default scene (subform) to use when viewing this object.
h_SetEditScene	The name of the default scene (subform) to use when editing this object.
h_PublishedVersionUNID	If this object is being edited and the current object is the draft version, the UNID of the published version of this object.
h_DraftVersionUNID	If this object is being edited and the current object is the published version, the UNID of the draft version of this object.

Field name	Description
h_LastTimePutAway	The last time that this object was changed: Published or Saved under construction.

Location fields

The fields listed in Table 41 are used to define Locations.

Table 41. Fields used to define Locations

Field name	Description
h_FolderUNID	The name or UNID of the Notes Folder where this page resides.
h_IsInToc	1 = This object is shown in the Table of Contents (sidebar).
h_CurrentPosition	The position of this object with respect to other objects in the collection.
h_SetParentUNID	If this is a child or response object, the UNID of the parent object

Security fields

The fields listed in Table 42 are used to define Security.

Table 42. Fields used to define Security

Field name	Description
h_Readers	If this object is protected from readership, the list of names, groups, and or roles that can read this object.
h_Authors	If this object is protected from authorship, the list of names, groups, and or roles that can author this object.

Workflow fields

The fields listed in Table 43 are used to define workflow status.

Table 43. Fields used to define Workflow

Field name	Description
h_WorkflowStage	Indicates the status of the document within the workflow. h_New = Created but not yet submitted for approval. h_Submitted = Has been submitted and it is being reviewed. h_Published = Has been approved. h_Rejected = Has been rejected.
h_SetNextStageUser	The name of the next person in the workflow cycle.

Field name	Description
h_CurrentApprover	A number designating the current person in the workflow cycle. 0 means the Originator. The list of persons associated with the workflow cycle is stored in the form used to create this page. (See h_Form).

Calendar fields

The fields listed in Table 44 are used to define Calendars.

Table 44. Fields used to define Calendars

Field name	Description
h_CalendarDate	The calendar fields are present when the page has been added to a calendar. The date that this object should appear in the calendar.
h_CalendarTime	The time that this object should appear in the calendar.
h_CurrentApprover	A number designating the current person in the workflow cycle. 0 means the Originator. The list of persons associated with the workflow cycle is stored in the form used to create this page. (See h_Form)
h_CalendarDuration	The length of time associated with this object when viewed in the calendar.

8.2.17 URL to create a Place

QuickPlace offers a very simple and powerful way to create a Place in one step by submitting a URL to the QuickPlace server. The arguments to use in this URL are:

```
http://server/
QuickPlace/QuickPlace/CreateHaiku.nsf?OpenDatabase&PresetFields=
h_SetEditCurrentScene;h_CreateManager,
h_EditAction;h_Next,
h_SetCommand;h_CreateOffice,
h_PlaceTypeName;PlaceTypeName,
h_Name;PlaceName,
h_UserName;UserName,
h_SetPassword;UserPassword,
h_EmailAddress;UserEmailAddress,
h_SetReturnUrl;ReturnURL,
h_OwnerAuth; AuthenticationType
```

Table 45. Arguments for creating QuickPlaces with a URL

Argument	Explanation
PlaceTypeName	The name of the PlaceType to use OR blank for the default PlaceType.
PlaceName	The name of the Place.
UserName	The Place's owner.
UserPassword	The Place's owner password.
UserEmailAddress	The Place's owner email address OR blank.
ReturnURL	The URL to go to after the Place is created; for example, the newly created Place.
AuthenticationType	Optional parameter. Specified how to autenticate the Place owner if an external directory is used. Value can be h_Local, h_External or h_Hybrid. if h_Hybrid is specified, an attempt will be made to use the external directory entry for the user name specified (if the passwords match). Otherwise a local user will be created.

8.2.17.1 Example

The following command provides an easy way to integrate the creation of Places into another application. We use it in 10.2, “A Domino application to handle requests for new Places” on page 281, and also discuss how you can use it in 10.4, “Integrating with Domino Workflow” on page 299.

```
http://mjollner.lotus.com/QuickPlace/QuickPlace/CreateHaiku.nsf?OpenDataba
se&PresetFields=h_SetEditCurrentScene;h_CreateManager,h_EditAction;h_Next,
h_SetCommand;h_CreateOffice,h_PlaceTypeName;,h_Name;millennia,h_UserName;N
ancy,h_SetPassword;opensesame,h_EmailAddress;nancy@yourdomain.com,h_SetRet
urnUrl;http://mjollner.lotus.com/millennia
```

8.2.18 Tools used to customize the QuickPlace Object Model

While much of a QuickPlace can be customized via a browser, there are some parts of QuickPlace which can only be customized using a Notes Client and or the Domino Designer.

Changes that can be made via a browser, using Web authoring tools such as an HTML editor, relate more to the user interface. For example, editing a layout file can be done using an HTML editor.

Changes made to QuickPlace Objects are done through the Notes client and in Domino Designer. For example, inspecting and customizing the images appearing in default QuickPlace pages must be done via the Notes Client.

In order to implement PlaceBots (Bots) in a QuickPlace, it is cumbersome to test the PlaceBots only in QuickPlace, especially when you are uploading your PlaceBots each time over the Internet. It is also possible that if you are running untested PlaceBots on a QuickPlace, you will crash the server. If you are testing locally in a Domino Designer test environment, you can use tools such as the debugger, which is not available via a Web browser.

We recommend using the Notes Client and Domino Designer as a local test environment. This allows you to quickly make changes on the PlaceBots, and provides you with an integrated development environment, with help files and debugging mechanisms. When you are finished creating and testing the PlaceBot in the Designer, upload it to the QuickPlace and finish the testing. Our experience has been that only finetuning changes need to be made in QuickPlace if the PlaceBot has been fully tested as a Domino Agent first.

Using the Notes Client and the Domino Designer also allows you to create new objects in a QuickPlace. Taking this a step further, it is possible to redefine the Object Model by adding your own features to a standard QuickPlace.

Core Domino technology is a proven platform, providing many powerful tools for a project. For developers who are not familiar with Domino, this means that they can use a wide range of development skills such as HTML, XML, DHTML, JavaScript, Image manipulation, Java and C++. Domino developers can also leverage Domino development skills, but either way it is possible to create robust Web-based applications, without having to “reinvent the wheel”.

If you are writing code to access QuickPlace Objects, we recommend that you use the Domino Designer.

8.2.19 QuickPlace Object Model and HTML: Building URLs

Building URLs in a QuickPlace is an important issue, due to the fact that QuickPlace is a browser-based application. Understanding QuickPlace URLs is also a good way of understanding the object hierarchy in QuickPlace.

The relationship between URLs and the QuickPlace Object Model flows in both directions. Understanding the structure of URLs helps us understand the QuickPlace Object Model. Conversely, once we understand the QuickPlace Object Model, we can use URLs to manipulate a QuickPlace.

URLs in QuickPlace use the same structure as in Domino. The following section explains how URLs are *built in a Domino Object Model*, and how these techniques are applied to the QuickPlace Object Model.

Domino URLs allow you to locate documents by using the key value of the first sorted column of a view, then generate a URL to link to a document using this key. Once the documents are located, they are not always opened in the browser. Sometimes they are read and their contents exposed and used by other Objects.

An example of when we want to locate a file without opening it is when a QuickPlace Theme layout file accesses a JavaScript LSS file. The user never sees the LSS page, but its contents are used by the visible page to render Objects and perform functions.

To locate a document in Domino, you must point the initial part of the URL to the host server, then to the database containing the required document. The next part of the URL must point to a view with the first column specified as being sorted. This first, sorted column becomes the key column. Then we use a URL to open the document:

```
http://Host/Database/View/Key?DominoURLCommand
```

Where:

View is the name of the view. To access a document regardless of the view, substitute a zero (0) for the view name and specify the document by its universal ID.

Key is the string, or key, that appears in the first sorted or categorized column of the view. If the key has spaces in it, substitute these for plus signs when creating a URL.

Use this syntax to open, edit, or delete documents and to open attached files. Domino returns the first document in the view whose column key exactly matches the Key.

There may be more than one matching document; Domino always returns the first match. The key must match completely for Domino to return the document. However, the match is not case-sensitive or accent-sensitive.

DominoURLCommand is the instruction to Domino of what to do with the file when found. For example, ?OpenDocument, ?EditDocument and ?DeleteDocument.

If this DominoURLCommand is omitted, a default will be substituted. For example, in the previous URL if you omitted the OpenDocument argument in a URL command,the document will still open because the command is automatically interpreted as OpenDocument.

As a rule, these URL Commands should always be used because we do not always want to open the file. Specific examples of these situations are provided later in this chapter.

The structure of URLs in a QuickPlace is the same as in any Domino database. QuickPlace Objects are often referred to via relative URLs. For example, to reference a page we have created, we use the following syntax:

```
../../h_View/PageName?OpenDocument
```

Where the "*../../*" section at the beginning of the URL creates a relative URL which is interpreted by the Domino server as referring to the parent objects of the current object (h_View and PageName).

Examples:

```
http://www.mercury.com/register.nsf/Registered+Users/Jay+Street?OpenDocument
http://www.mercury.com/register.nsf/0/466c5172561e1c5c852566c2005f6bbb?OpenDocument
```

Note: Many QuickPlace Objects in QuickPlace have internal names beginning with h_. This refers to the internal name of QuickPlace which is Haiku.

To reference images, JavaScript library files, or files other than pages, the following syntax can be used:

```
../../h_Index/Document+Name/$File/Imagename.gif?OpenElement
```

Or you can use:

```
../../h_Index/Document+Name/$File/ScriptLibName.js?OpenElement
```

Note: Many objects in QuickPlace can be located via the h_Index View. It contains links to many of the published objects in a QuickPlace.

Note: When referencing a JavaScript file, the ?OpenElement argument should always be used. This is to tell Domino that the file being accessed is not a page to open, which is the default action.

For more information on Domino URLs, look at the Domino Designer help file under the heading Domino URL Commands. In these documents are complete explanations of the syntax structure.

8.2.20 Building URLs: Referencing images

The following section deals with the issue of using images in QuickPlace. Due to the fact that QuickPlace is a platform for creating Web sites, images form a vital part of the QuickPlace Object Model.

Fortunately, QuickPlace's structure provides many ways to include images in your pages. For example, when creating Theme layout files, the images are automatically uploaded into the QuickPlace when the layout file is uploaded. This section explains the small amount of preparation required to uploading images for Theme layout files.

This section is designed to help you include images using simple techniques that you will use most of the time. It also shows how to include image resources in situations where you have created highly customized QuickPlace. In other words, it describes the basics, and gives you the tools to create highly customized solutions.

Note: This section does not describe manipulating the default QuickPlace images, for example the Go.gif image that appears next to the simple search field. These topics are handled in 8.3, "Editing QuickPlace Objects directly with the Notes Client" on page 259.

Note: This section also does not describe techniques involved where fully automated importing procedures exist within QuickPlace. An example of automated importing is when you create and upload a Microsoft Word file. When you do this, the images are imported without any interventions. This section does not deal with images in Word files, it only deals with instances where some developer intervention is required (for example when you are creating a Theme layout file, or writing an importable HTML file, or referencing files required to display the results of PlaceBot and so on). Creating layout files are described in Chapter 4, "Creating Themes" on page 47.

You can import images in the following ways:

Method 1

Provide a URL to an image and let QuickPlace upload the image for you.

- Use this when creating layout files s and imported HTML documents that do not use JavaScript to reference images.

Method 2
Create a URL, have QuickPlace upload it, then reference it using HTML or JavaScript

- Use this when rendering an image using JavaScript.

Method 3
Manually upload images into a document and reference them via URLs from a separate document.

- Use this method if the image is very large and you want users' browsers to be able to cache the image.
- Or use this method if the image is referenced within a JavaScript function (QuickPlace does not import images when they appear within JavaScripts).
- Or use this method if the image is referenced within a PlaceBot which creates new pages.

Your solutions may be a mixture of all three.

Referencing images: Method 1
- Create a Theme layout file or HTML Imported page
- Let QuickPlace import it for you.
- This works in Theme layout files and imported HTML.

This is the easiest way of importing images together with Theme layout files and HTML pages. When you create a valid link to an image within an HTML page or a Theme layout file, QuickPlace will upload it for you automatically when you upload the Theme layout file or HTML file. The only technical issue you need to understand to get this to work is how to create a valid URL.

1. In your Theme layout file or importable HTML document, download all the images in a local directory. The simplest way to do this is to save them in the same folder as the Theme layout file or HTML page. We saved ours in the same folder, so the URL we used in the HTML file was:

   ```
   <img src="transparent.gif" width=5 height=1 alt="" border="0">
   ```

   Note: You are not forced to have all files in the same directory. You just have to make sure that you have a valid reference to your images. For example, if you place all image files in a directory named *images* under the directory where your HTML files are, the URL to reference your image just has to start like this:

   ```
   <img src="images/transparent.gif" width=5 height=1 alt="" border="0">
   ```

2. If you are editing a QuickPlace Theme layout file that you downloaded from a QuickPlace version 2.0 QuickPlace, you will need to make some changes to the structure of the images in the Theme layout files.

   For more information on downloading the default QuickPlace Themes, see Chapter 4, "Creating Themes" on page 47.

   This change is to work around a problem in QuickPlace 2.0, which stops images in default Theme layout files from appearing if the Theme layout files have been downloaded from a QuickPlace. This will be fixed in coming releases, and can be quickly worked around by following these steps:

   a. Make sure that all references to images are removed that have the following structure:

      ```
      img src="../../../$resources.nsf/h_ResourcesByName/transparent.gif/ $FILE/
      transparent.gif?OpenElement"
      ```

   b. Change them to this structure:

      ```
      img src="transparent.gif"
      ```

   c. The SRC for the image should be a valid URL. A valid URL is a URL that successfully references a file. The test for a valid URL is listed in the final step of this section.

   d. Download the images and make sure that the images are in the same folder as the Theme layout file you are writing.

3. If you cannot find the transparent.gif which is used to space elements in the page, you can make your own 10 x 10 pixel totally transparent gif, or reuse the ecblank.gif that can be found in the domino/icons directory.

4. Upload the Theme layout file or Import the file, and QuickPlace will automatically import all the referenced files. Instructions on uploading files are included in the section on Themes.

5. Take the following steps to test the importing of image files:

   a. Create an image into your Theme layout file or HTML file in an open area of the file. An open area of an HTML page is an area that is not nested in a table or a QuickPlaceSkinComponent.

      The reason for putting the image in an open area of the form is because if you are editing a Theme layout file, other QuickPlace components being rendered above the image can interfere with the SRC structure and stop the image from being rendered properly.

   b. Preview the file in your browser. If it shows up, it will be uploaded by QuickPlace.

c. Remove the image from the document if you do not intend to have it there permanently.

Referencing Images: Method 2

- Prompt QuickPlace to import the file into the current document.
- Reference the file using JavaScript.
- This method is the most efficient method to use when you are referencing an image via a JavaScript function (in a Theme layout file, for example) or when updating a document via a PlaceBot. It is possible to use Method 3 when referencing images in JavaScript, but Method 3 is more complex, for the reasons specified in the Method 3 description.

To force QuickPlace to upload the image, you just need to create a valid URL to the image at the top of the page.

This presents us with a problem: we are rendering an image in a position where we do not want to see it. Commenting the image out will not work, because QuickPlace then ignores the image altogether.

To make this work, we render it in a in a 1 pixel x 1 pixel size. By making it too small to see, the image is still uploaded, but the user will not notice the image. In order for this to work, the images must also be named when they appear in their 1x1 pixel format. By using this name, we can make the image available to JavaScripts below it on the page. This will not work if the image is referenced in JavaScript above the images on the page.

To do this, simply add the image to the source code near the top of the page. We started the tag like this:

```
<img src="mouseOverON.gif"
```

Then, so that we could reference it in our JavaScript, we named the img:

```
name=imgON
```

So that the image was not visible, we shrunk it down to be only one pixel wide and one pixel high.

If we reference images just in JavaScript, QuickPlace will not upload the image. The way to force QuickPlace to upload the image is to render it in HTML format, then reference the uploaded image in the JavaScript.

Note: This rendered image must appear above the JavaScript referencing it. The syntax for sizing it is:

```
border=0 height=1 width=1>
```

Now that this image has been named, it is possible to reference it in your JavaScript code...

```
...
document.imageInDocument.src = document.mouseOverON.src
...
```

Example

The following text is based on the script in our Millennia example which we describe in Chapter 4, “Creating Themes” on page 47:

First of all we define two images right at the top of the page:

```
<img name=GetImage1Off src=Delete.gif width=1 height=1>
<img name=GetImage1On src=DeleteHiLite.gif width=1 height=1>
```

Then, further down in the script of the page, the JavaScript declares two new variables as images. Once they are declared, the script assigns an SRC to them, using the SRC from the images written in the page above.

Note: Do not use any other images named with the same name or the JavaScript will become confused and will not display the image.

Note: Quickplace will have processed the images previously described during the import process and edited their SRC paths to be correct within the QuickPlace. If you write an explicit URL to the images, they will work on your local copy of the QuickPlace, but not when other people are using the QuickPlace.

The following code shows the lines in a JavaScript tag that implement this technique of referencing images based on named images:

```
<script language="JavaScript">
   img1off = new Image();
   img1off.src = GetImage1Off.src;
   img1on = new Image();
   img1on.src = GetImage1On.src;
```

For an example of this technique in the context of a full HTML document, see 4.4, “Adding the Favorites HTML page” on page 104.

Next we deal with the fnImageON and fnImageOFF functions, applying the new SRC to the image being moused-over, or moused-out:

```
   function img_on(imgName) {
      imgOn = eval(imgName + "on.src");
      document [imgName].src = imgOn;
   };
       function img_off(imgName) {
       imgOff = eval(imgName + "off.src");
       document [imgName].src = imgOff;
   };
```

In the main body of the page, the images appear with the mouseOver and mouseOut events referenced:

```
<img name="img1" src="Delete.gif" width=23 height=23 border=0 alt="Magic"
   onmouseover="img_on('img1')" onmouseout="img_off('img1')">
```

Here is the page in its entirety:.

```
<HTML>
<HEAD>
<TITLE></TITLE>
<META name="description" content="">
<META name="keywords" content="">
</HEAD>
<BODY BGCOLOR=FFFFFF TEXT=000000 LINK=0000FF VLINK=800080>
<!-- ----===oooOoo=======
   ' Simple Mouse Over File
   ' -------=oooOoo===---- -->

<img name=GetImage1Off src=Delete.gif width=1 height=1>
<img name=GetImage1On src=DeleteHiLite.gif width=1 height=1>

<script language="JavaScript">
   img1off = new Image();
   img1off.src = GetImage1Off.src;
   img1on = new Image();
   img1on.src = GetImage1On.src;

   function img_on(imgName) {
      imgOn = eval(imgName + "on.src");
      document [imgName].src = imgOn;
   };
       function img_off(imgName) {
       imgOff = eval(imgName + "off.src");
       document [imgName].src = imgOff;
   };
</script>
<img name="img1" src="Delete.gif" width=23 height=23 border=0 alt="Magic"
   onmouseover="img_on('img1')" onmouseout="img_off('img1')">
</BODY>
</HTML>
```

This file is available among the downloadable files that come with this Redbook.See Appendix L, “Additional Web material” on page 425 for instructions on how to get the sample.

Referencing Images: Method 3

- Import file into a document used specifically for holding resources and reference it via a URL.
- This method is more complex than Method 2, but using it is necessary when you are automatically creating new documents (for example, via a PlaceBot), and the file does not already exist in the document context.
- It is a good technique to know because the same procedure can be used for many different file types. For example, you can reference Java classes,

Shockwave files or any other type of file that can be referenced in an HTML file.

To use images referenced from another document, take the following steps:

- Create an image and name it; we called ours imgLogo.gif.
- Open your browser, go to your QuickPlace and create a new document in that QuickPlace; we named ours *ResourcesDoc*.

  Do not use spaces in resources you access in browsers. This file will not be seen by most users, so the readability of its name is not important. If you must use spaces, remember to replace the spaces with plus (+) signs when creating URLs.

- Import the GIF into the ResourcesDoc, via the upload control.
- Save the resource document.
- In the object you want to reference this image from (Page, Theme layout file, PlaceBot and so on), create a tag referencing the image. Ours looked like this:

```
<img src=../../h_Index/ResourcesDoc/$File/imgLogo.gif?OpenElement>
```

- Always appending the ?OpenElement string to the URL is a good technique, because Domino requires this for some file types such as attachments and OLE objects. Image files do not require this argument, but it is a good habit to have because many other types require the argument in order for Domino to access them correctly.
- To test the document's URL, make sure that you have your browser open to your QuickPlace.
- Go back into your HTML file, and copy everything after the ../../ of your IMG SRC to the clipboard, shown here in bold:

<img src=../../**h_Index/ResourcesDoc/$File/imgLogo.gif?OpenElement**>

- Return to your browser, and make sure that your QuickPlace is still open. Paste the text into the URL bar, after /main.nsf/; your URL should look something like this:

```
http://server.com/myQuickPlace/main.nsf/h_Index/ResourcesDoc/$File/imgLogo.gif?OpenE
lement
```

- Check to see that the image appears. If you get it wrong, you will get an `Error 404 - File Not Found` message from your browser.

8.2.21 Building URLs: referencing JavaScript library files

JavaScript libraries are used by many JavaScript programmers to centrally store functions, and declare variables so that they can easily be reused in many places. We find that this technique saves development time and reduces the likelihood of errors.

Another significant advantage of using JavaScript libraries is that they are rarely need to be changed, because they contain programming logic, not content. This means that the JS file can be cached on the server. Even when a readable Web page referencing the library is modified and needs to be reloaded from the server, the cached library file can be reused.

The tradeoff with using JavaScript Libraries is that they exist as a separate file, which must be referenced as an element. The following section describes how this can be done in a QuickPlace.

In a Web site where HTML files can be easily located via hardcoded URLs, you can reference a JS file simply with a reference that looks like this:

```
<script src="scripts.js"></script>
```

After this file has been located, all the functions and variables in this library can be used in the page. In QuickPlace we are able to do exactly the same thing, but calculating the SRC path to the file is a little more complicated.

To use JavaScript libraries, you must create a JS file and import it as an attachment. Then use the Domino syntax structure to reference the file:

```
../../view/document/$File/attachmentname.filetype?OpenElement
```

In the following example we import a JavaScript Library file, but the basics of this technique can be used for importing other files such as Java Classes, WAV file, Shockwave files or OLE objects.

When you reference the JS file, you need to first reference the document containing the file, then the file itself. For more information look at the Domino Designer help file under the heading Domino URL Commands or Appendix D, “Domino URL commands” on page 371.

Here is how to reference a JavaScript Library File in a QuickPlace:

- Write the file and name it (we called ours ScriptLib.js).
- Create a new document in a QuickPlace (we named ours ScriptsDoc*)*. While it is possible to use names here that contain spaces, it is easier not to.

  This file will not be used by most users, so its appearance is not that important. If you do use spaces, remember to replace the spaces with plus (+) signs when creating URLs.
- In the object (Page, Theme layout file, PlaceBot and so on), create a tag referencing the JS file. Ours looked like this:
  ```
  <script language=JavaScript
     src=../../h_Index/ScriptsDoc/$File/scriptLib.js?OpenElement>
  ```

```
</script>
```

Note: Remember to use the </script> tag, or it will not work.

- Make sure that you append the ?OpenElement to the URL or it will not work.
- Test the URL by copying everything after the ../../, up to and including the ?OpenElement command, to the clipboard, and pasting it into the URL bar after /main.nsf/.

  You will get an option box appear asking you if you want to download the file. If you get an error from your browser, make sure that you have followed all steps described above. When we did this, our URL looked like this:

  ```
  http://servername.com/QuickPlace/quickplace/Main.nsf/h_index/ScriptsDoc
  /$File/scriptLib.js?OpenElement
  ```

Note: As stated above it is important to append the ?OpenElement to the end of your URL or the file may not be properly retrieved. This is because Domino will try to open the element as the default element type, which treats the element as a page.

To trap errors that occur when the browser attempts to use functions and variables in the library file, we declared a new variable with a name not existing in the user document (the readable file accessing the JS file). We put in the following line in the ScriptLib.JS file:

```
var ping = 1;
```

Then in the user document we put in the following line:

```
if ( typeof ping == 'undefined' ) {
    document.write( ‘File Not found’);
} else {
    doOtherStuff...
```

If the user document can access the JS file, the variable will be declared in the user document, and the functions and variables in the library file can be used in the page. If the JS file is not found, this method makes it possible for the JavaScript in the user document to work around any potential errors that will otherwise occur.

If you create the ScriptsDoc in the main Room of your QuickPlace, you now have a JS library file available to all user documents in the QuickPlace.

Note: Users accessing this library document must have reader access to the document. This can be a problem when a JavaScript library file is attached to a document in a Subroom and the user file is located in the main Room of the

QuickPlace. The error will occur when a user has access to the main Room but not the Subroom containing the document. This happens because QuickPlace uses reader field security on documents, and therefore on attachments to those documents.

We recommend that Pages used to hold attachments are always created in the main Room. This way, the JS library will work, even when new rooms are created.

This technique can be used for any sort of attachment file types.

8.2.22 Structural elements

Most pages that we see in a QuickPlace are created using the h_PageUI form. This special form is used to create a range of different content pages by having different values saved in different fields.

For example, a key field in the h_PageUI form is the h_Type field. If this field has the value of “0”, the document will present itself as a QuickPlace Page. If the h_Type field has a value of “1”, the document will present itself to the user as a QuickPlace Folder. For more information, see 8.2.16, “Common QuickPlace Object fields” on page 240.

In QuickPlace, nomenclature works in the following way. JavaScript variables in a QuickPlace page have exactly the same names as the Fields. For example, the h_Type field’s value is available in the JavaScript variable called h_Type. The h_ prefix stands for Haiku and denotes the field as containing information used internally to a QuickPlace. They should be considered read-only, as modifying them in pages may cause errors in the code of the page.

Fields (and therefore Javascript variables) that begin with c_ are custom fields. They are more often able to be manipulated without causing errors. We recommend that you do not manipulate these variables, but instead create new variables based on the QuickPlace variables and modify your new values.

It is not possible to document these c_ (custom) fields because they do not belong to the core product and may vary from release to release of QuickPlace. We recommend that when you create your own Domino Fields or JavaScript variables, you prefix them with a character. We prefixed ours with m_ for Millennia.

8.3 Editing QuickPlace Objects directly with the Notes Client

Many of the Objects used in QuickPlaces can be edited to customize a QuickPlace. Some of the resources are stored in the individual QuickPlaces. Some resources are stored in the resources.nsf database. This section focuses on editing the resources.nsf database, but the techniques here can be applied to the individual QuickPlaces.

This example assumes that you have access to the server directory structure and can access the server console. You also need the Lotus Notes and Domino Designer R5.0 Client.

8.3.0.1 Editing the simple search button image

In the following example, we edit the image that appears next to the search field in Pages.

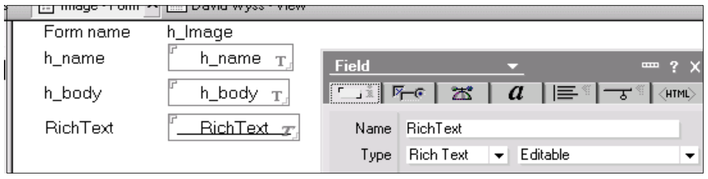
Identifying the image in your Web browser

1. The first step in making changes to elements is knowing what you are looking for. With images, this is fairly straightforward. Go to any page that contains the image. This can be in the opening page of any Place that uses the default Theme.
2. Right-click the image and select **Save Picture As...**.
3. Save the file, and remember the name that appears in the File Name field. The name of the file used for the search field is “Go” , and the file type is GIF, so we know we are looking for Go.gif.
4. Now that we know the name of the image, we need to find it in the QuickPlace databases. If the image is used in many places, such as the Go.gif, is it probably in the resorces.nsf QuickPlace database.
5. Find the image you saved and make the changes you want using a graphic tool like PaintShop Pro. Leave the image in the same dimensions and file name (Go) and file type (GIF) as it was.

Creating a Form to allow editing

1. Make sure the QuickPlace server is shut down.
2. Open the QuickPlace resources.nsf database in Domino Designer.
3. Create a new Form.
4. Create a new field on the Form called RichText. Make the Type “Rich Text”.
5. Go to Form Properties in the properties floating window.
6. Make the form the Default Database Form.

7. Name the form “Image”.
8. (Optional) Create an editable Text field called h_Name.
9. (Optional) Create an editable Text field called h_Body.



Form name h_Image
h_name h_name
h_body h_body
RichText RichText

Name RichText
Type Rich Text Editable

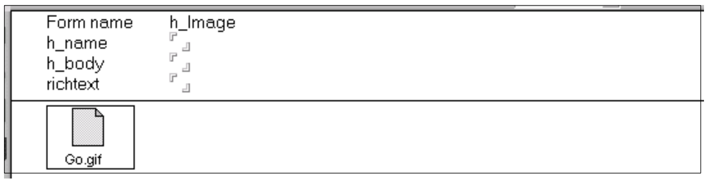
Figure 87. The Image Form

10.Save the Form; your Form should look similar to Figure 87.

Opening the image in the Notes Client.

1. Make sure the QuickPlace server is shut down.
2. Open Lotus Notes.
3. Go to the Notes menu and choose **File -> Database -> Open...** Or press Control + O.
4. Press the browse button and locate resources.nsf. We ran QuickPlace on top of Domino and we found it on:
   `C:\Lotus\Domino\Data\QuickPlace\resources.nsf`
5. If you are running QuickPlace as a standalone server, you will find it in a path like this:

   `C:\QuickPlace\Data\resources.nsf`
6. Open the database. If you cannot open the database, switch your ID to the ID of the server used on the QuickPlace.
7. Open the view named **hResourcesByName**.
8. Scroll down to “go.gif” and open it by double-clicking it.
9. If you get an error message that a Form cannot be displayed, simply click OK.
10. The document will open and you will see the Go.gif in the document as an attachment; see Figure 88 on page 261.



Form name	h_Image
h_name	◸ ◿
h_body	◸ ◿
richtext	◸ ◿

Figure 88. The Go.gif Image selected in the Image form

11.Put the document into edit mode by double-clicking a blank part of the Form.

12.Select the image attachment and press the Delete key.

You will see a warning. Press **OK** to accept deleting the attachment.

13.Click in the *RichText* Field and import the image you changed.

14.Save the document.

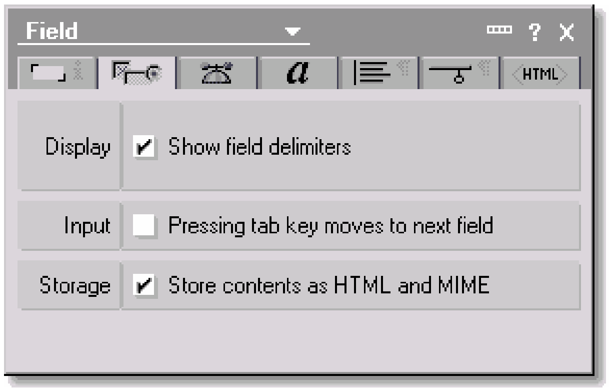
The graphic used with the search box will now be the one you specified for all Places on the QuickPlace server.

Note: This change will be overwritten if you upgrade your QuickPlace server to a higher level.

8.4 The MIME Field Type in Domino Designer

The MIME field type used internally in QuickPlace is a field type based on the RichText field type in Domino. The MIME (which means Multipurpose Internet Mail Extensions) field type allows you to attach non-text files to Internet mail messages. Non-text files include graphics, spreadsheets, formatted word-processor documents, and sound files.

To create a MIME field in Domino Designer, create a RichText field type and then click Store contents as HTML and MIME, in the Field Properties; refer to Figure 89.



Display ✓ Show field delimiters

Input Pressing tab key moves to next field

Storage ✓ Store contents as HTML and MIME

Figure 89. Selecting the HTML and MIME option in a RichText field's properties

8.5 The tools we used

You can use many different tools to customize QuickPlace. In theory, most of what we describe in this book can be accomplished using a simple text editor and Domino Designer. However, in reality you will want to use dedicated tools like HTML editors, Java development environments, and so on.

In this section, we describe the tools we used for our examples. This is not an endorsement of these tools, it is simply a list of the tools we used.

TopStyle by Bradsoft
This is a CSS StyleSheet editor with a couple of particularly useful features. It shows you a list of possible properties that you can suggest, then allows you to select a relevant value for that property. It helps you create cross-browser style sheets by showing you if that property is supported in different browsers and browser versions.

Cute HTML by GlobalSCAPE
This is a simple, robust HTML editor that allows you to hand-code HTML files. Other products that are useful for creating HTML files add elements in automatically, whereas Cute HTML allows you a bit more control. The snippets menu allows you to quickly add new chunks of code.

Lotus Domino Designer R5
We used this to develop all our Java and LotusScript PlaceBots. We also used it to mimic a live QuickPlace site, where we ran our PlaceBot testing. We found that testing in QuickPlace was sometimes too cumbersome.

Microsoft Internet Explorer 5.0, Microsoft Internet Explorer 5.5
These are the standard browsers for uploading and manipulating QuickPlaces.

Netscape Navigator 4.6, Netscape Navigator 4.72
These are the standard IBM browsers for testing the functionality of QuickPlaces.

NotesPeek 1.52 for R5
NotesPeek presents the information in Domino databases as it is available through the Lotus API. We used it, for example, to see the values of internal QuickPlace fields. NotesPeek is available as a free download from:

`http://notes.net/sandbox.nsf`

We also used various image editing programs.

8.6 Summary

In this chapter we have described the QuickPlace Object Model. We have described many of the internal fields and their values to help you understand the internals of QuickPlace.

We have also provided examples of how to access the object from your code, and discussed programming considerations such as how to reference images, work with JavaScript libraries, and so on.

Chapter 9. QuickPlace Developer’s Kit and the event interface

We have already discussed powerful things that can be accomplished by customizing QuickPlace though Themes, Forms, PlaceBots and so on. However, you can go even further and do some amazing things together with the QuickPlace Developer’s Kit.

In this chapter we will give an overview of the QuickPlace Developer’s Kit (QDK) and then focus on the support in the toolkit for extending QuickPlace commands through the event interface.

9.1 What is the Lotus QuickPlace Developer's Kit

The QDK is a downloadable collection of files that includes:

- Documentation describing how to customize a QuickPlace by directly modifying the QuickPlace objects
- Documentation describing how to use PlaceBots, Themes/Skins and PlaceTypes
- Sample agents for creating QuickPlace members
- A framework for extending the QuickPlace commands with a user-developed DLL

At the time of writing, the QDK still needed to have some documentation added and we worked with a beta release. You can download the QDK by going to the QuickPlace DevZone at the following Web address:

```
http://www.quickplace.com/devzone
```

The QuickPlace developers also post other programming samples and utilities in the DevZone. In Appendix C, “QuickPlace utility programs” on page 369, we include a brief description of two useful utilities from the DevZone.

In this chapter, we focus on how the QDK allows you to extend QuickPlace commands by utilizing the QuickPlace event interface.

Note: While the QDK allows you to do some really powerful things, it also requires you to work with core QuickPlace objects. The data schema described in the QDK is subject to change in future versions of the QuickPlace product. Applications written to this data schema may need to be modified in order to work with future versions of the product. Keep this in mind as you read the rest of this chapter.

9.2 Using the QuickPlace event interface

You can capture certain QuickPlace commands and attach your own logic (written in C or C++) to run when a QuickPlace command is executed. For example, you can write a function that executes each time the CreateRoom command is called and places an entry in a log about who created a room, when is was done, and in which QuickPlace it happened.

To use the QuickPlace event interface, you first need to have the Lotus C API Toolkit for Domino and Notes (formerly known as the Notes API) installed. You can download the Lotus C API Toolkit by going to the Technologies and Products section on Lotus Developer Network at this Web address:

```
http://www.lotus.com/developer
```

The Lotus C API Toolkit works with Microsoft Visual C++ Development System for Windows. You can also use the IBM VisualAge C++ toolkit, but it has not been tested thoroughly and there are some limitations; see the *Lotus C API User Guide* (included with the download) for more information.

Install the Lotus C API. Set up Path, Lib and Include statements for the environment, and test the installation by compiling the *introwin* program as described in the user's guide in the document named *Building 32-bit Windows Applications*.

You also need the include file and library files supplied with the QDK, and then you are ready to extend QuickPlace. In your code, you register a hook for the event you want to capture, indicating whether you want your code to run *before* or *after* QuickPlace has processed the command. You compile your code into a DLL and place it in the QuickPlace program directory. Finally, you stop QuickPlace and add the following line to NOTES.INI:

```
QuickPlaceModules=yourmodulename.dll
```

where *yourmodulename.dll* is the name of your DLL. You can specify several extension modules in the same line, separating them with a comma like this:

```
QuickPlaceModules=yourmodulename.dll,yourmodulename2.dll
```

Restart QuickPlace, and your extension code is running.

9.2.1 Events are captured for all QuickPlaces on the server

When you write a DLL where you specify you want to capture an event to do your own processing, be aware that *all events of the specified type* will be trapped in your code. If you only want to do special processing on one QuickPlace or a specific group of QuickPlaces, you have to test for this in your code. The fact that event trapping is server-wide also means that you have to consider what impact on overall server performance your code may have.

9.2.2 The events listed in the QPHook sample

The list of events that you can capture are shown in the QDK sample program QPHook. You can start off your own program from the QPHook program by commenting out the events you do not want to catch and adding your code to the event(s) you want to trap. Table 46 shows the events that were listed in the QPHook sample program at the time of writing.

Table 46. QuickPlace events that can be extended using the QDK

QuickPlace events	
h_ChangeAcl	h_EditIndex
h_CheckHuName	h_ExternalDirectory
h_CheckMemberName	h_ForceSSL
h_ConfigureMailSettings	h_GetSkinGroupName
h_CopyAgent	h_LookupForCreate
h_CopyForm	h_LookupMember
h_CopyMoveFolder	h_MoveCopyPages
h_CreateAgent	h_MoveRoom
h_CreateFolder	h_Page
h_CreateGroup	h_PreProcessForm
h_CreateOffice	h_PublishForm
h_CreatePlace	h_PublishPlaceType
h_CreateRoom	h_PublishSkinGroup
h_CreateUsers	h_RefreshPlaceFromPlaceType
h_DeleteAgent	h_RefreshPlaceTypesGallery
h_DeleteAttachments	h_ReorderItems
h_DeleteFolder	h_ReorderRoomArea
h_DeleteForm	h_RunAgent
h_DeleteGroups	h_RunAgentOnPublish
h_DeleteOffice	h_SendMail
h_DeletePages	h_ServerBasics
h_DeletePlaceType	h_SetAesthetic
h_DeleteRoom	h_SetNewUsersAllowed
h_DeleteSkinGroup	h_SetNoteProperties
h_DeleteUsers	h_SpellCheck
h_EditAgent	h_UpdateGroup
h_EditFolder	h_UpdateUser

The events are not fully documented in the QDK 2.0 beta, but for many of the events, you can see from their name which command they trigger.

Later in this chapter we discuss how you can get the data note associated with the events and thus see which fields and values are passed for each event.

9.2.3 The sample event interface programs in the QDK

Besides the QPHook sample, two other sample event interface programs are included in the QDK. They are:

- QPAudit - shows how to track creation of new rooms and log it to a Domino database.
- QPCustomInvite - shows how to pick up the text for invitation letters sent to new members from a Domino database.

Also there are other sample programs worth mentioning available on the DevZone in the Goodies:Samples section:

- QPLegal - shows how to add text (for example a legal disclaimer) to e-mail notifications being sent from the QuickPlace server.
- DevCon 2000 Samples - contains various sample command extension programs; for example, how to switch theme, based on who the user is.

Note: Besides invites and notifications, newsletters are also being sent out by e-mail. However, there is currently no way to alter the newsletter text.

QPCustomInvite and QPLegal can actually be used “as is” to make e-mail invites and e-mail notifications going out from your QuickPlace server contain standard text that you choose. We use these two add-in DLLs when we pull our full solution together and build a TurnKey Server in 11.6, “Deploying your customized QuickPlace as a Turnkey Server” on page 340, but we will make one small change.

The texts to include in the e-mails are stored in Domino databases on the QuickPlace server. Somebody might be tempted to try and hack into them and change to the e-mails being sent. Therefore, we will use database names different from the published sample databases. In the next section, we will go through the changes necessary to the QPCustomInvite program.

9.3 Modifying the QPCustomInvite program

In this section we go through the steps needed to make a very simple change to the sample program QPCustomInvite that is coming with the QDK. The program reads text strings to include in r-mail invitations from a Domino database named *qpcustdoc.nsf*. We will instead have the program read from a database named *redqptxt.nsf*.

9.3.1 Prerequisites

You need the following tools to modify the QDK samples:

- Lotus C API for Domino and Notes

  Available from the Technologies and Products section on:

  `http://www.lotus.com/developer`

  Installation and setup is described in the user's guide that is part of the download. The QDK samples expect the API to be installed in a directory named **\NotesAPI** on the same drive as the QDK samples are installed on.

- Microsoft Visual C++ Version 6.0

  It is part of Microsoft Visual Studio. You can install it, accepting all default choices.

- QuickPlace Developer's Toolkit 2.0 Beta

  Available from the QuickPlace DevZone at:

  `http://www.quickplace.com/devzone`

  We installed it under the directory D:\QDK.

- Updated QPCustomInvite sample

  An updated version of the QPCustomInvite sample is available at the QuickPlace DevZone in the Home:Goodies:Samples section. We replaced the original QPCustomInvite sample with this new version.

9.3.2 Doing the modification

We are now ready to change the program. The QDK sample programs comes as C source programs and Microsoft Visual C++ project files where the paths have been set up relatively. This means that you do not have to change anything in the project setup, as long as the include file and the library files coming with the QDK are in directories on the same level as the directory with the sample program. Do the following:

1. Start the Window Explorer and open the directory with the QPCustomInvite source files. In our case, the path was:

   ```
   D:\QDK\qdk2.0beta\Samples\QPCustomInvite
   ```

2. Find the Microsoft Visual C++ project workspace file named **QPCustomInvite.dsw** and double-click on it to launch Microsoft Visual C++.

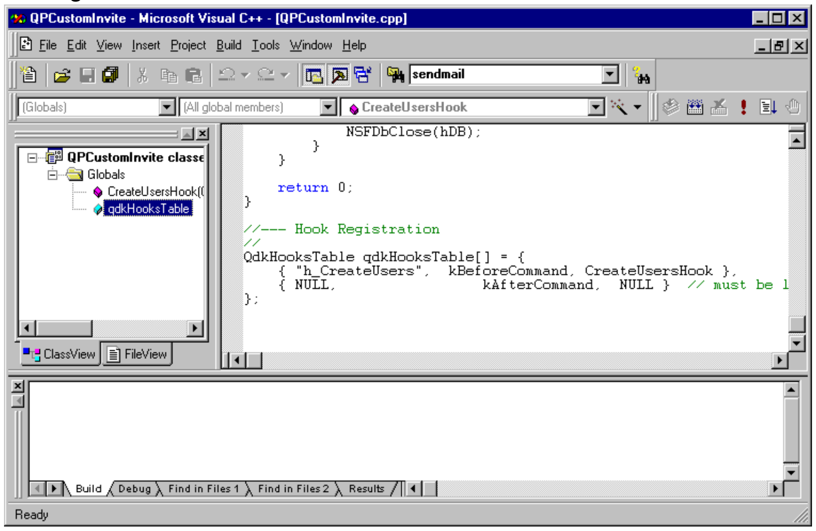
3. In the left pane of Microsoft Visual C++, click the plus sign next to the item named **QPCustomInvite classes**.

   - A folder named **Globals** should appear. Expand that as well.

   You should now see two entries named (or starting with):

   - CreateUsersHook(
   - qdkHooksTable

   Double-click **qdkHooksTable**. Your window should now look similar to Figure 90.



```
                NSFDbClose(hDB);
            }
        }

        return 0;
}

//--- Hook Registration
//
QdkHooksTable qdkHooksTable[] = {
    { "h_CreateUsers",  kBeforeCommand, CreateUsersHook },
    { NULL,                      kAfterCommand,  NULL }  // must be l
};
```

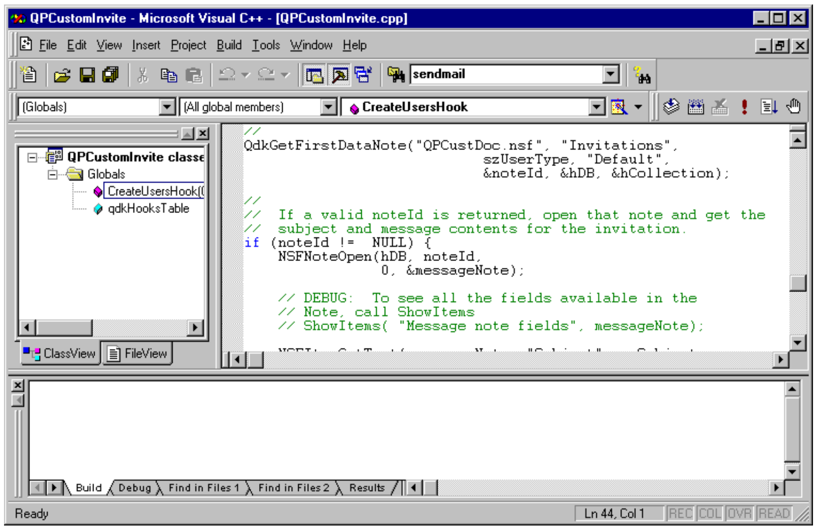
Figure 90. qdkHooksTable in Microsoft Visual C++

The following code is listed in the big pane:

```
QdkHooksTable qdkHooksTable[] = {
    { "h_CreateUsers",kBeforeCommand,CreateUsersHook },
    { NULL,            kAfterCommand,NULL }// must be last
};
```

This is where we tell the QuickPlace server that we want to extend the QuickPlace command named *h_CreateUsers*. The parameter *kBeforeCommand* means that we want our code to run *before* QuickPlace does its own processing on the command, and finally there is the name of the function to call in our code when the command is triggered which is *CreateUsersHook*. We will now walk briefly through that code.

4. In the left pane of Microsoft Visual C++, double-click on the item starting with **CreateUsersHook**. Your window should now look similar to Figure 91 on page 271.



```
//
QdkGetFirstDataNote("QPCustDoc.nsf", "Invitations",
                            szUserType, "Default",
                            &noteId, &hDB, &hCollection);

//
//  If a valid noteId is returned, open that note and get the
//  subject and message contents for the invitation.
if (noteId !=  NULL) {
    NSFNoteOpen(hDB, noteId,
                0, &messageNote);

    // DEBUG:  To see all the fields available in the
    // Note, call ShowItems
    // ShowItems( "Message note fields", messageNote);
```

Figure 91. CreateUsersHook in Microsoft Visual C++

Scroll through the code in the big pane in the right side of the window. Documentation is added as code comments. The major things happening in the code are as follows:

- A data note is associated with the event. It holds the information necessary to send out an e-mail invitation to a new user. The code gets the value of the field in the data note named *h_SecurityAction*.

  The value of this field indicates whether the newly created user has right as administrator, author or reader.

  This value is used to determine what kind of customized message we want to send to the new user.

- Using QDK function called *QdkGetFirstDataNote*, we open the Domino database where we have stored our customized welcome messages. Some of the parameters to this function are:

  • Name of Domino database: *QPCustDoc.nsf* (this is what we will change)

  • Which view to look in: *Invitations*

  • Key to look up (type of user): *h_AddManager*, *h_AddAuthor* or *h_AddReader*

  • Default key value to look up if primary key isn't found: *Default*

- Once the right document has been retrieved from the Domino database, we read the contents of the two fields *Subject* and *Message* from that document.
- Next, these values are written to the data note that is associated with the h_CreateUsers event. The subject is written to the field named *h_SetPublishEmailSubject*, and the message is written to the field named *h_SetPublishEmailMessage*.
- Finally the function ends with a return value of zero to the QuickPlace server.

5. We will now do our simple code modification. Find the QdkGetFirstDataNote function and change the Domino database name from *QPCustDoc.nsf* to **RedQPTxt.nsf.**
6. We compile our DLL right away. The project is set up to generate a DLL linked with a debug library, but for this simple change, we will create a DLL without debug code in it right away. Do do this, we first need to switch our active configuration in Microsoft Visual C++.

   Select **Build -> Set Active Configuration...**
7. Select *QPCustomInvite - Win32 Release* and click **OK**.
8. Build the DLL by selecting **Build -> Build QPCustumInvite.dll**.

   Verify in the status pane in the lower part of the window that the DLL file was built with zero errors and zero warnings.

   If no errors or warning occurred, the DLL is ready to be deployed on the QuickPlace server. The DLL file is in a directory named *Release* under the one where our sample code is.
9. Copy the *QPCustomInvite.dll* file to the QuickPlace server program directory.
10. Copy the Domino database named *qpcustdoc.nsf* to the QuickPlace server data directory with a new name of **redqptxt.nsf**.

    Change the access control list (ACL) of the database to make sure it only can be changed by the administrators of your QuickPlace server.
11. Stop the QuickPlace server. Add the following line to the NOTES.INI file on the QuickPlace server program directory:

    ```
    QuickPlaceModules=qpcustominvite.dll
    ```
12. Restart the QuickPlace server. You should now be able to change to content of your e-mail invitation for your QuickPlace server by editing the documents in the RedQPTxt.nsf database.

Using the same procedure, we also modified the *QPLegal* sample program to read the mail notification prolog from a database named *RedQPLegal.nsf*.

9.4 How to figure out which fields to modify

In the walkthrough of the QPCustomInvite program in the previous section, we saw that two fields in the data note passed together with the event were changed:

- h_SetPublishEmailSubject
- h_SetPublishEmailMessage

We also needed to know in which field QuickPlace stores the value for type of user (h_SecurityAction) and which values that field could have (h_AddManager, h_AddAuthor or h_AddReader).

You may wonder how to figure out the names of the fields you want to work with. In our case it is pretty simple because the field names were supplied to us in a sample program, but you probably run out of useful sample programs before you get to do everything you may want to do when extending the QuickPlace commands.

The QDK toolkit has a utility function that can help you.

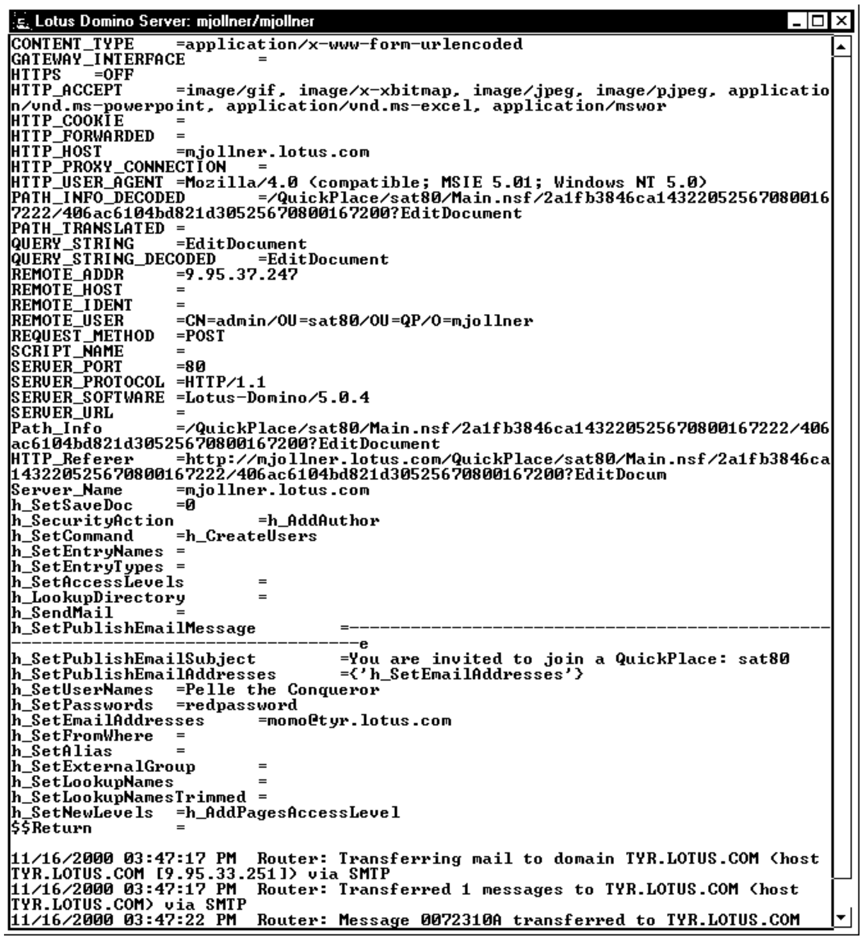
9.4.1 Using the QdkShowItems function to list fields

There is a help function in the QDK that list all fields (items) in a data note on the server console. We added the following line to our QPCustomInvite code before doing any changes to the data note:

```
QdkShowItems( "====>  In CreateUsersHook", hNote);
```

Note: If you are running QuickPlace as a standalone server, you need to stop it in the Services panel and start the server manually to be able to see the server console.

In Figure 92 on page 274, you can see the output we got on our server console when using the QPCustomInvite.dll with QdkShowItems enabled.



```
CONTENT_TYPE    =application/x-www-form-urlencoded
GATEWAY_INTERFACE       =
HTTPS   =OFF
HTTP_ACCEPT     =image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, applicatio
n/vnd.ms-powerpoint, application/vnd.ms-excel, application/mswor
HTTP_COOKIE     =
HTTP_FORWARDED  =
HTTP_HOST       =mjollner.lotus.com
HTTP_PROXY_CONNECTION   =
HTTP_USER_AGENT =Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
PATH_INFO_DECODED       =/QuickPlace/sat80/Main.nsf/2a1fb3846ca14322052567080016
7222/406ac6104bd821d30525670800167200?EditDocument
PATH_TRANSLATED =
QUERY_STRING    =EditDocument
QUERY_STRING_DECODED    =EditDocument
REMOTE_ADDR     =9.95.37.247
REMOTE_HOST     =
REMOTE_IDENT    =
REMOTE_USER     =CN=admin/OU=sat80/OU=QP/O=mjollner
REQUEST_METHOD  =POST
SCRIPT_NAME     =
SERVER_PORT     =80
SERVER_PROTOCOL =HTTP/1.1
SERVER_SOFTWARE =Lotus-Domino/5.0.4
SERVER_URL      =
Path_Info       =/QuickPlace/sat80/Main.nsf/2a1fb3846ca14322052567080016 7222/406
ac6104bd821d30525670800167200?EditDocument
HTTP_Referer    =http://mjollner.lotus.com/QuickPlace/sat80/Main.nsf/2a1fb3846ca
14322052567080016 7222/406ac6104bd821d30525670800167200?EditDocum
Server_Name     =mjollner.lotus.com
h_SetSaveDoc    =0
h_SecurityAction        =h_AddAuthor
h_SetCommand    =h_CreateUsers
h_SetEntryNames =
h_SetEntryTypes =
h_SetAccessLevels       =
h_LookupDirectory       =
h_SendMail      =
h_SetPublishEmailMessage        =------------------------------------------------
----------------------------------e
h_SetPublishEmailSubject        =You are invited to join a QuickPlace: sat80
h_SetPublishEmailAddresses      ={'h_SetEmailAddresses'}
h_SetUserNames  =Pelle the Conqueror
h_SetPasswords  =redpassword
h_SetEmailAddresses     =momo@tyr.lotus.com
h_SetFromWhere  =
h_SetAlias      =
h_SetExternalGroup      =
h_SetLookupNames        =
h_SetLookupNamesTrimmed =
h_SetNewLevels  =h_AddPagesAccessLevel
$$Return        =

11/16/2000 03:47:17 PM  Router: Transferring mail to domain TYR.LOTUS.COM (host
TYR.LOTUS.COM [9.95.33.251]) via SMTP
11/16/2000 03:47:17 PM  Router: Transferred 1 messages to TYR.LOTUS.COM (host
TYR.LOTUS.COM) via SMTP
11/16/2000 03:47:22 PM  Router: Message 0072310A transferred to TYR.LOTUS.COM
```

Figure 92. Output from QdkShowItems in QPCustomInvite.dll

Here you can see all fields available to you exactly when your code starts to extend the command to create a user.

For more information about the QDK as it progresses, check the QuickPlace DevZone at:

`http://www.quickplace.com/devzone`

9.5 Summary

In this chapter we have given you an overview of the QuickPlace Developer's Kit 2.0 Beta (QDK). We briefly described what it contains and then looked at the QuickPlace event interface that allows you to extend QuickPlace commands through C/C++ programs. This is a very powerful feature that unfortunately isn't fully documented at the time of writing. Therefore we also discussed ways for you to figure out what data you can capture and potentially modify when you register to extend one of the QuickPlace events.

Keep in mind that the QuickPlace data scheme is subject to change, so you may have to update programs using the event interface when upgrading to future versions of QuickPlace.

Chapter 10. Integrating with other applications

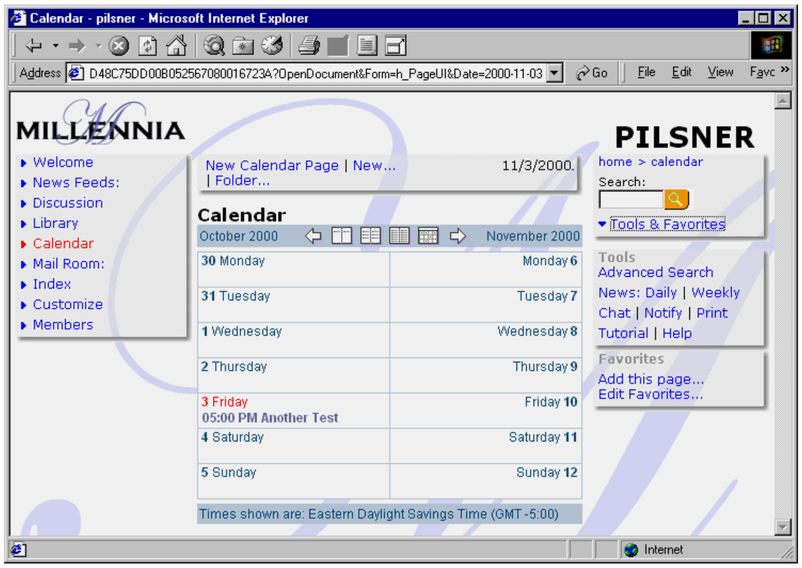
In this chapter we show examples of how QuickPlace can integrate with other applications. We discuss and show examples of how to:

- Access data in QuickPlace using Java (from a Notes client)
- Enhance QuickPlace with a Domino Web application
- Archive documents in Domino.Doc
- Initiate jobs in Domino Workflow from QuickPlace
- Integrate QuickPlace in a portal

10.1 Mirroring data to Notes - a Java agent example

In this example we show you how to create a Java agent for the Notes client to mirror a QuickPlace calendar to your personal Notes calendar. Any event in the QuickPlace calendar will show up in your personal Notes calendar as well.

You can see an example of how the agent works in Figure 93, where we first see an entry in the QuickPlace calendar, and Figure 94 on page 278, where this entry has been copied to the Notes calendar by our agent.



Calendar

October 2000	November 2000
30 Monday	Monday 6
31 Tuesday	Tuesday 7
1 Wednesday	Wednesday 8
2 Thursday	Thursday 9
3 Friday 05:00 PM Another Test	Friday 10
4 Saturday	Saturday 11
5 Sunday	Sunday 12

Times shown are: Eastern Daylight Savings Time (GMT -5:00)

Figure 93. QuickPlace calendar

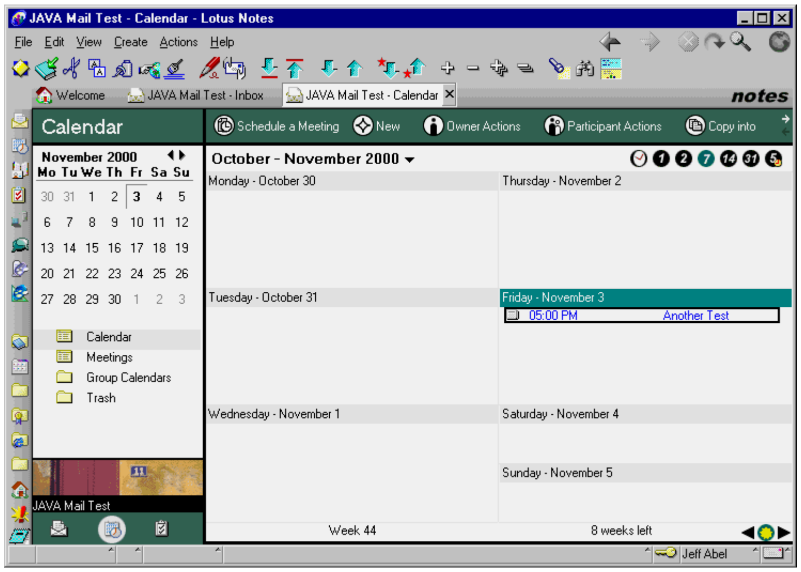


Figure 94. The QuickPlace calendar entry in the Notes calendar

Note: This example is from the DevCon 2000 examples at the QuickPlace DevZone. It is only meant to illustrate techniques, which means there are limitations you would not have in a production application.Some of the limitations are that the QuickPlace server name is hardcoded, and the sample code only creates Appointment-type entries (AppointmentType=4).

The following is a discussion of code samples from the Java agent that we will be using. The complete agent can be downloaded from the IBM Redbooks Web site. See Appendix L, “Additional Web material” on page 425 for instructions on how to get the source file.

First we will instantiate a new Notes Session object and create an agent log to log actions to. Here is the code to get this done. You will call the log.logAction() method to log actions.

```
Session session = getSession();
AgentContext agentContext = session.getAgentContext();
Agent agent = agentContext.getCurrentAgent( );

Log log = session.createLog( "QpEventSync Agent Log");
log.openAgentLog();
log.logAction( "----------------------------------------");
```

Now we need to establish the QuickPlace with which we will synch our calendar. This is done by opening the current database and the QuickPlace room database using the following:

```
String serverName= "lager/lager";
String qpName= "pilsner";
String qpRoom= "Main.nsf";

// open this database
Database db = agentContext.getCurrentDatabase( );
log.logAction( "opened database " + db.getTitle( )  + "    [" + db.getFileName( ) + "]");

// connect to the quickplace server
DbDirectory dir = session.getDbDirectory( serverName);  // quickplace server
log.logAction( "connected to server " + dir.getName( ) );

// open quickplace database
Database QPdb =dir.openDatabase( "QuickPlace/" + qpName + "/" + qpRoom);  // quickplace
name
log.logAction( "opened database " + QPdb.getTitle( )  + "    [" + QPdb.getFileName( ) +
"]");
```

Next we need to open the calendar view of both databases as shown.

```
// open the calendar view in the qp db
View QPview = QPdb.getView( "h_Calendar");   // qp calendar view name
log.logAction( "opened view " + QPview.getName( ) );

// open this db's calendar view
View view = db.getView( "Calendar");   // calendar view name
log.logAction( "opened view " + view.getName( ) );
```

We will then loop through each document in the calendar view of the QuickPlace to find new or modified entries. When looping through the documents in the calendar view, each date interval of a repeating event is treated as a separate document. Therefore, we need to check whether we have already processed the first document in the date list iteration. Otherwise, we will be creating a new document for each date iteration of the repeating event. The following code is used to accomplish this test.

```
if ( QPdoc.getUniversalID( ).compareTo( previousUNID) != 0) {
log.logAction( ">");
log.logAction ( "processing document:  " + QPdoc.getItemValueString("h_Name") + "   UNID:
" + QPdoc.getUniversalID( ) ) ;

// if document already exists, then update it, otherwise create a new one
// searches for the document using the QPref_UNID field and create a temp view with the
search results
int docFound = view.FTSearch( QPdoc.getUniversalID( ), 1);

// update the document in this database
if ( docFound == 1) {
log.logAction ( "   document exists in db.  updating...") ;
doc = view.getFirstDocument( );

// create an empty document in this database
} else {
log.logAction ( "   document does not exists in db.  adding...") ;
doc = db.createDocument( );
```

Now we add some information to this document to tell us later that this document came from the QuickPlace and was not originally created by us in our calendar.

```
// Record this document's origin
// set QPref_UNID equal to the quickplace doc UNID so that we can later know if the doc
was already downloaded
doc.replaceItemValue("origin_QPdocUNID",  QPdoc.getUniversalID( ) );
doc.replaceItemValue("origin_QPserver",serverName);
doc.replaceItemValue("origin_QPname",qpName);
doc.replaceItemValue("origin_QProom",qpRoom);
```

We now add the content from the QuickPlace document to the new or existing document in our calendar.

```
Item item = QPdoc.getFirstItem("h_Originator");
doc.replaceItemValue("From", item.getValueString( ) );

item = QPdoc.getFirstItem("h_Name");
doc.replaceItemValue( "Subject",  item.getValueString( ) );

item = QPdoc.getFirstItem( "h_DateTimeList");
doc.replaceItemValue("CalendarDateTime", item.getValues() );

// if this event is an "all day event", do not add a StartDateTime item to the new doc
if ( QPdoc.getItemValueString( "h_AllDayEvent").compareTo( "1") != 0) {
    item = QPdoc.getFirstItem( "h_DateTimeList");
    doc.replaceItemValue("StartDateTime", item.getDateTimeValue( ) );
    }

// event repeats
if ( QPdoc.getItemValueString( "h_Repeats").compareTo( "1") == 0) {
     doc.replaceItemValue("Repeats", QPdoc.getItemValueString( "h_Repeats") );
     }

item = QPdoc.getFirstItem( "h_CalendarDate");
doc.replaceItemValue("StartDate", item.getDateTimeValue( ) );

item = QPdoc.getFirstItem( "h_CalendarTime");
doc.replaceItemValue("StartTime", item.getDateTimeValue( ) );

item = doc.replaceItemValue("AppointmentType", "4" );
item = doc.replaceItemValue("Form", "Appointment" );

// Copy the PageBody item because the item is of mime type
if ( !QPdoc.hasItem( "$FILE" ) ) {
    if ( docFound == 1) {
        doc.removeItem( "Body"); // remove if updating an existing doc. otherwise, copy
will add another copy.
        }
    item = QPdoc.getFirstItem( "PageBody");
    item.copyItemToDocument(doc, "Body");
    }
```

Lastly, we save this document and then continue looping through the remaining documents in the QuickPlace calendar view.

```
// save the new document
doc.save();
log.logAction( "   document saved...  UNID: " + doc.getUniversalID( ) ) ;
}
```

Once we finish looping through all of the documents, we need to close the log that we created.

```
log.logAction( "--------------------------------------");
log.close();
```

That's it--you should now have all the calendar entries in your calendar that the QuickPlace calendar has. We have seen how, using Java, you can access the QuickPlace calendar from another application (in our example, a Domino database). In this case, data is going from one application (QuickPlace) to another (Notes). In the next section we look at an example where we use a Domino database to add functionality to our QuickPlace server.

10.2 A Domino application to handle requests for new Places

On a default QuickPlace server installation, everyone is allowed to create Places. However, in many cases there is a requirement to "filter" who can create Places (for example, to make sure that user departments get charged for their use of the QuickPlace server, or to verify that the Place is created for business use or for some other reason).

In this section we show you a skeleton Domino application designed for such "filtering" of QuickPlace requests. In this example we focus on proving the concept of handling request input and approval in our Domino database, and on showing how the look and feel of the application can be set to match our standard QuickPlace Theme (in our case, the Millennia Theme).

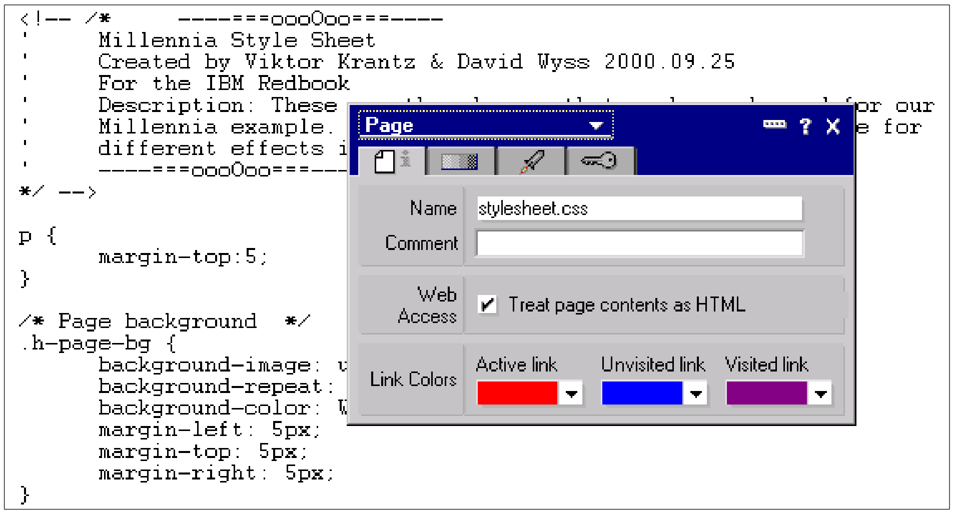
This is an example you can build on; it is *not* ready for production without further work. For example, the request application does not check whether a Place already exists with the name a requestor inputs. Also, the Access Control List, together with a little application modification, must be set up to allow users to enter requests, while only a certain group or role must be allowed to approve requests.

In QuickPlace, you can generate and configure a Place with a single URL command, as described in 8.2.17, "URL to create a Place" on page 244. We utilize this URL command in our request application. Based on the data input from the user, we construct the URL and have the QuickPlace server create the new Place when we approve the request.

The sample database can be downloaded from the IBM Redbooks Web site. See Appendix L, "Additional Web material" on page 425 for instructions on how to get it.

10.2.1 Making the Domino application look like a QuickPlace Theme

All Pages and Forms we created in the application use the same HTML, Style Sheet and JavaScript as in the regular Millennia Theme. First we copied all the text in the Millennia Style Sheet into a new Domino Page, named the page *stylesheet.css*, and made sure that the Web Access property was set to "Treat web content as HTML".



```
<!-- /*      -----===oooOoo=======
'       Millennia Style Sheet
'       Created by Viktor Krantz & David Wyss 2000.09.25
'       For the IBM Redbook
'       Description: These                              for our
'       Millennia example.                              e for
'       different effects i
'       ----===oooOoo===---
*/ -->

p {
        margin-top:5;
}

/* Page background  */
.h-page-bg {
        background-image: u
        background-repeat:
        background-color: W
        margin-left: 5px;
        margin-top: 5px;
        margin-right: 5px;
}
```

Name: stylesheet.css
Comment:
Web Access: ✓ Treat page contents as HTML
Link Colors: Active link 🟥 Unvisited link 🟦 Visited link 🟪

Figure 95. The Millennia Style Sheet inside a page

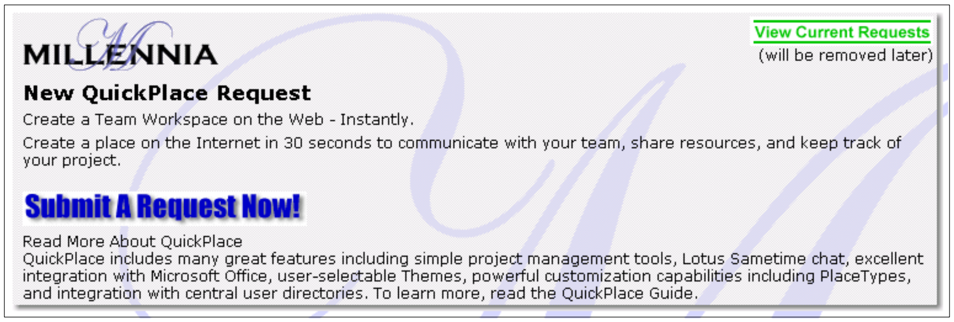
Then we created the Domino Pages *Main*, *qprequested* and *qprequests* and the Domino Form *QuickPlace Request* with HTML from the Millennia Theme to get the 3-D look we wanted. We made sure that all HTML was marked as *Pass-Thru HTML* and we put formulas in the "HTML Head Content" Object to prevent showing old cached data and to include the Style Sheet.

```
"<META HTTP-EQUIV=\"cache-control\" CONTENT=\"no-cache\">" + @Char(0) +
"<META HTTP-EQUIV=\"Pragma\" CONTENT=\"no-cache\">" + @Char(0) +
"<META HTTP-EQUIV=\"Expires\" CONTENT=\"0\">" + @Char(0) +
"<LINK REL=\"stylesheet\" HREF=\"/" +
@ReplaceSubstring(@Subset(@DbName;-1);"\\";"/") + "/stylesheet.css\"
TYPE=\"text/css\">" + @Char(0)
```

We included the class *h-page-bg* in the "HTML Body Attributes" Object and we put some JavaScript in the "JS Header" Object.

10.2.2 The request and approval logic

On the *Main* Page, we put two images with Hotspot Resource Links to the *QuickPlace Request* Form and to the qprequests page. The link to *qprequests* is a link that can be set to be visible only for managers using a hide-when formula when using the application in production (we have not done this in our application).



New QuickPlace Request

Create a Team Workspace on the Web - Instantly.

Create a place on the Internet in 30 seconds to communicate with your team, share resources, and keep track of your project.

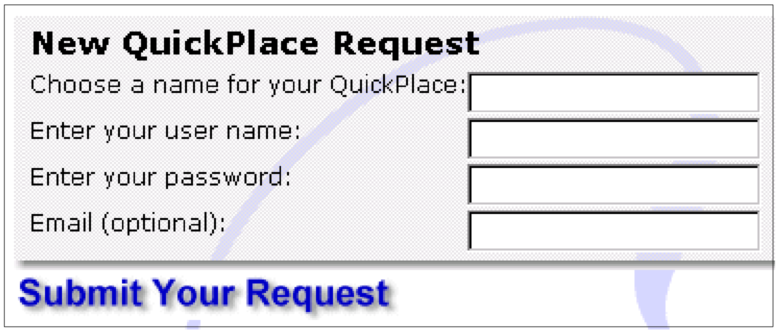
Read More About QuickPlace

QuickPlace includes many great features including simple project management tools, Lotus Sametime chat, excellent integration with Microsoft Office, user-selectable Themes, powerful customization capabilities including PlaceTypes, and integration with central user directories. To learn more, read the QuickPlace Guide.

Figure 96. QPFront home page with link to view with all requests showing

The *QuickPlace Request Form* has the fields that the requester fills in, together with two images with @Formulas. The first image *Submit your Request* is only shown to the user if the document hasn't been saved. This is accomplished using a hide-when formula in Domino. When the user clicks *Submit Your Request*, the document is saved and a page named *qprerequested* (which thanks the user for the request) is returned. The code for this is shown here:

```
@Command([FileSave]) ;
@Command([FileCloseWindow]) ;
@URLOpen("/qpfront.nsf/qprequested?OpenPage")
```



New QuickPlace Request

Choose a name for your QuickPlace:	
Enter your user name:	
Enter your password:	
Email (optional):	

Figure 97. QuickPlace Request Form

The second image on the page, named *Approve And Create Now*, has the following @Function:

```
qpreturnURL := qpSrv + "/qpfront.nsf/qprequests?OpenPage" ;
urlstring :=
"/QuickPlace/QuickPlace/CreateHaiku.nsf?OpenDatabase&PresetFields=h_SetEdi
tCurrentScene;h_CreateManager,h_EditAction;h_Next,h_SetCommand;h_CreateOff
ice,h_PlaceTypeName;,h_Name;" + qpname + ",h_UserName;" + qpmgrname +
",h_SetPassword;" + qpmgrpword + ",h_EmailAddress;" + qpmgremail +
",h_SetReturnUrl;" + qpreturnURL ;

@SetField("qpCreated" ; "1") ;
@SetField("dsp_qpurl" ; qpSrv + "/" + qpname);
@PostedCommand([FileSave]) ;
@URLOpen( urlstring )
```

The field qpSrv is set when the user creates the request. We utilize the CGI variable HTTP_referer coming from the browser to extract the server name. Here is the code to do this:

@SetField("qpSrv"; @LeftBack(@LeftBack(HTTP_Referer; "/"); "/"))

This will take a URL like the following:

```
http://mjollner.lotus.com/qpfront.nsf/qpRequest?OpenForm
```

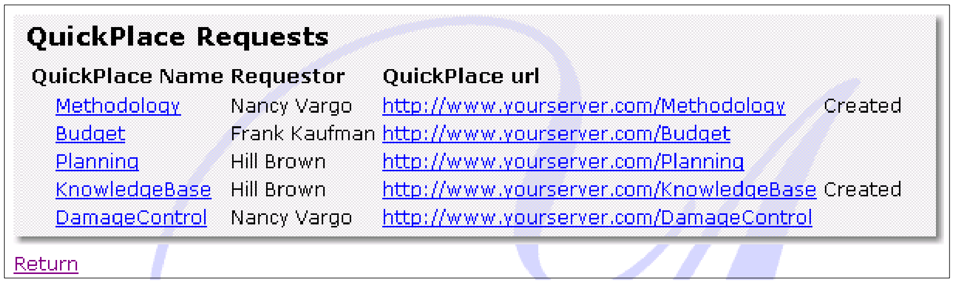
and strip it down to this:

```
http://mjollner.lotus.com
```

Note: This code assumes that qpfront.nsf is in the data directory of the QuickPlace server because it simply cuts everything off from the rightmost slash character twice. It has to be modified to work if the database is in a subdirectory.

For an explanation of the Create QuickPlace URL syntax, see 8.2.17, “URL to create a Place” on page 244.

The qprequest Page has the only view in the database embedded into it showing Managers both the documents that are up for approval, and those that have been approved.



QuickPlace Requests

QuickPlace Name	Requestor	QuickPlace url	
Methodology	Nancy Vargo	http://www.yourserver.com/Methodology	Created
Budget	Frank Kaufman	http://www.yourserver.com/Budget	
Planning	Hill Brown	http://www.yourserver.com/Planning	
KnowledgeBase	Hill Brown	http://www.yourserver.com/KnowledgeBase	Created
DamageControl	Nancy Vargo	http://www.yourserver.com/DamageControl	

Return

Figure 98. The QPFront view of all requests

Obviously the application can offer more functionality, with Cost Center fields in the request form and agents that send e-mail to managers and requesters, but this example shows you the basic concept of how to create such an application.

10.2.3 Integrating the request application with QuickPlace

There are many scenarios for adding our Domino request database to a QuickPlace server. We discuss two ways:

- Let the request application page be the default home page for the QuickPlace server.
- Modify the QuickPlace Welcome page to point to the request database.

10.2.3.1 Changing the default server home page

To set our request page in the Domino database as the default server page, we need to modify the names.nsf database in the server's data directory (it is the Domino Directory, if you have an overlay installation). The easiest way to do this is using a Notes client, but you can also write a small Domino API program to do it (for example, a VisualBasic program that uses the Domino COM interface).

The steps for using a Notes client are:

1. Open names.nsf in the server's data directory.
2. Open the view Server -> Servers.
3. Locate the server document for your server.

   If you have a standalone installation, there should only be one document in the view.
4. Open the server document in edit mode (Ctrl+E) and click **Internet Protocols**.

5. In the Mapping section, there is a field named Home URL with a value of */homepage.nsf?Open* (for overlay installation) or */QuickPlace* (for standalone installation. Replace the value of the Home URL field with:

   ```
   /qpfront.nsf
   ```

6. Save the server document, close names.nsf, and you are done.

With these changes, users that simply type in the address of the server will now see the request application page as shown in Figure 96 on page 283.

10.2.3.2 Modifying the QuickPlace Welcome page

We also tried to modify the standard QuickPlace Welcome page so that when users click *Create a QuickPlace Now!* they are shown the request application page from our Domino application instead of the default QuickPlace page. We used Domino Designer, the Notes client and an HTML editor.

Note: This is an undocumented and thus unsupported procedure. We only describe what we did and what worked for us, but you should be aware that this procedure may not work in all cases.

The Welcome page is attached as an HTML file to a document in the main.nsf database of the administrator's Place on the server. We detached the HTML file, modified it, and replaced the attached file with our new version. However, first we had to copy a Domino form to the database in order to be able to open the document in the Notes client.

We did the following to modify the Welcome page:

1. In Domino Designer, open main.nsf for the QuickPlace administrator's Place; in our case, the path was:

   ```
   C:\Lotus\QuickPlace\data\QuickPlace\quickplace\main.nsf
   ```

   This database has no Domino forms, so we need to copy a page form from a main.nsf database in another Place.

2. Open a main.nsf database from another Place on the server. We opened:

   ```
   C:\Lotus\QuickPlace\data\QuickPlace\methodology\main.nsf
   ```

3. In Domino Designer, select the Page form (the alias is QDK_h_Page) and copy it to the clipboard.

4. Select the QuickPlace main.nsf. Click the **Forms** view and paste the Page form into it.

   Now we have what we need to get to the welcome page. First we must locate the document where it is attached.

5. Click the **Views** view for the QuickPlace main.nsf in Domino Designer and select (do not open) the **System\QDK** view (the alias is h_QDK).
6. Open the view *in the Notes client* by selecting **Design -> Preview in Notes**.
7. Find the document with the title **Welcome**. Titles are listed in the third column of the view. In our case, the document was listed after *Folder Instructions* and *Examples*.

   Open the document and detach the file named **setupWelcome.htm**.
8. Open setupWelcome.htm in your HTML editor.

   Find this string:

   ```
   href="../../../CreateHaiku.nsf?OpenDatabase"
   ```

   Replace it with the string to open the Domino request database. In our case, we used:

   ```
   href="/qpfront.nsf/qprequestFromMain?openform"
   ```

   Save the changed setupWelcome.htm.
9. In the Notes client, put the Welcome document in edit mode (if you haven't already done so):
   - Select the setupWelcome attachment and press the **Delete** key.

     Confirm that you want to delete the attachment.
   - Place the cursor in the **PageBody** field and attach the modified setupWelcome.htm file using **File -> Attach**.
   - Save the document and close the main.nsf database.

The next time users click the create a QuickPlace link on the welcome page, they will be directed to the Domino request application.

By modifying the welcome page instead of making the Domino request page the default home page, your users will still have access to links like *Read More About QuickPlace*, *QuickPlace Guide* and *Examples*. However, you will have to accept the default QuickPlace look for the welcome page unless you want to dig deeper into QuickPlace internals.

10.3 Integrating with Domino.Doc

QuickPlace has simple document management capabilities built in. But sometimes, there is a need for a more robust system built for document management to help manage the content created in QuickPlace.

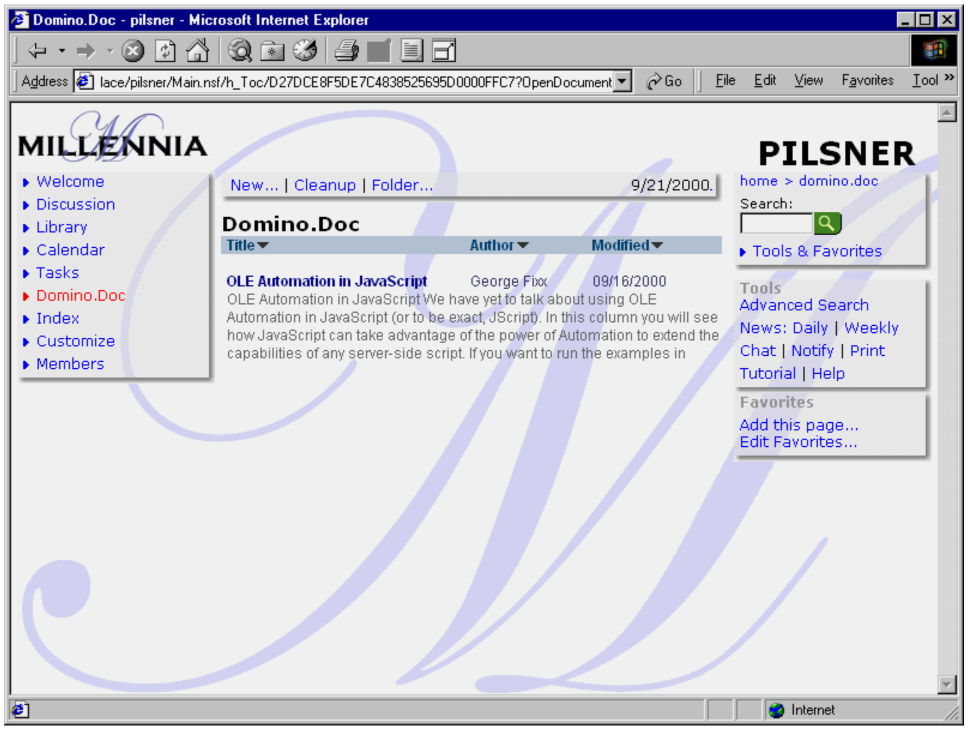
Content creation in QuickPlace is an “organic” model suitable for teaming and, while it has rudimentary version control, it could benefit from a document management system such as Domino.Doc. Domino.Doc provides a structured storage model that is much more suitable for life cycle management of content.

We will illustrate, through the use of an example, how to reap the benefits of integrating QuickPlace with Domino.Doc by leveraging the strengths of each product. The example will show you how to effortlessly move selected content that was created in QuickPlace to a Domino.Doc Library so that the content can be benefit from the management facilities in Domino.Doc.

The complete sample can be downloaded from the IBM Redbooks Web site. See Appendix L, “Additional Web material” on page 425 for instructions on how to get the sample files including the Domino.Doc library and file cabinet templates with the modifications we have added to them.

10.3.1 Overview of the example

This example integrates QuickPlace with Domino.Doc by adding functionality to QuickPlace and Domino.Doc that allows a user to have content, which was created in QuickPlace, stored in a Domino.Doc Library. This custom functionality is in the form of a special Domino.Doc folder along with a PlaceBot that will send pages in this folder to the Domino.Doc Library. The user will either copy or move pages to this folder to have them stored in Domino.Doc.



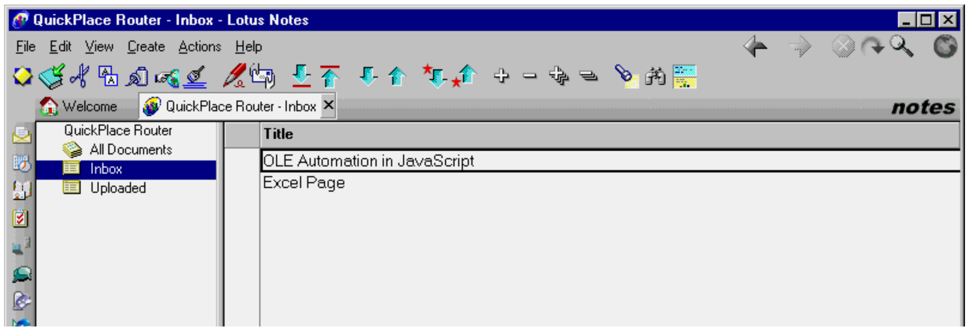
Domino.Doc

Title	Author	Modified
OLE Automation in JavaScript	George Fixx	09/16/2000

OLE Automation in JavaScript We have yet to talk about using OLE Automation in JavaScript (or to be exact, JScript). In this column you will see how JavaScript can take advantage of the power of Automation to extend the capabilities of any server-side script. If you want to run the examples in

Figure 99. Domino.Doc folder in QuickPlace

Once the user has placed a page in the Domino.Doc folder, a PlaceBot named the *QPRouter* will send the page contents to a Domino database on the Domino.Doc server named the *QPRouter Processor*. Here, the page contents will be uploaded to the Domino.Doc library on a scheduled basis.


QuickPlace Router - Inbox - Lotus Notes
File Edit View Create Actions Help
Welcome
QuickPlace Router - Inbox
notes
QuickPlace Router
All Documents
Inbox
Uploaded

Title
OLE Automation in JavaScript
Excel Page

Figure 100. QPRouter Processor with pages waiting to be uploaded

After the QPRouter Processor uploads the page contents to the Domino.Doc library, the user that initiated the action to store the page in Domino.Doc will be notified that the page contents have been successfully stored in the Domino.Doc library.

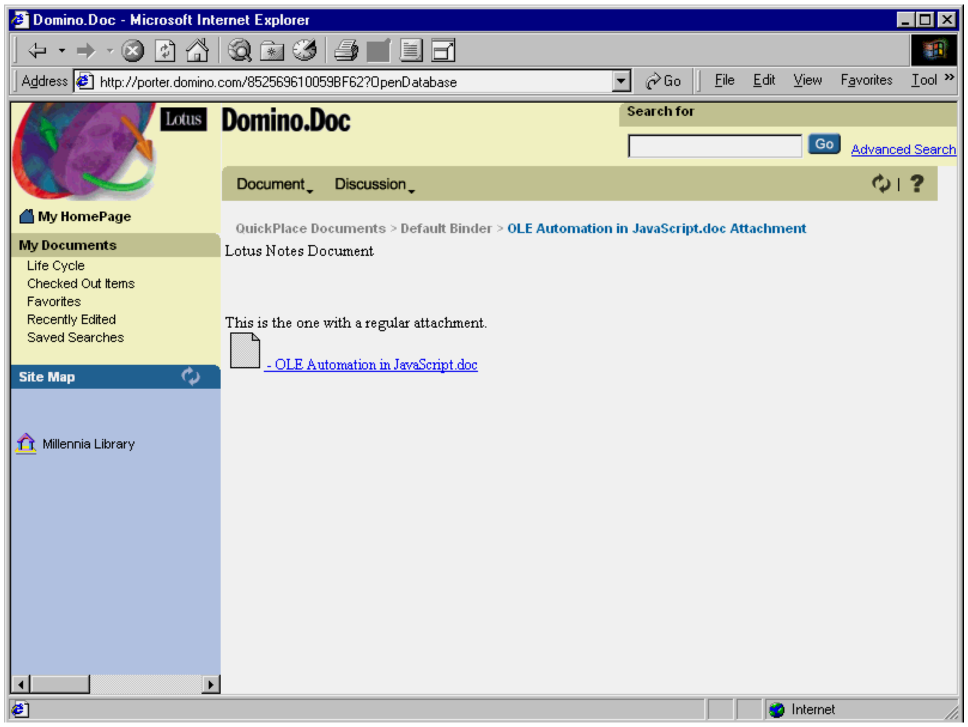

Domino.Doc - Microsoft Internet Explorer
Address http://porter.domino.com/852569610059BF62?OpenDatabase
Domino.Doc
Search for
Go
Advanced Search
Document
Discussion
My HomePage
My Documents
Life Cycle
Checked Out Items
Favorites
Recently Edited
Saved Searches
Site Map
Millennia Library
QuickPlace Documents > Default Binder > OLE Automation in JavaScript.doc Attachment
Lotus Notes Document
This is the one with a regular attachment.
- OLE Automation in JavaScript.doc
Internet

Figure 101. Page content as it now exists in the Domino.Doc library

The following is a list of areas which you should be familiar with in order to set up the example.

QuickPlace requirements

- QuickPlace Server Setup on an Existing Domino Server
- Creating a QuickPlace
- Creating Folders in QuickPlace
- Creating PlaceBots in QuickPlace

Domino.Doc requirements

- Domino.Doc Server Installation and Library Setup
- Creating a custom Document Type in Domino.Doc

- Creating a File Cabinet in Domino.Doc
- Using the Domino.Doc API

Domino Designer requirements

- Creating Forms, Subforms, and Views
- Writing LotusScript Agents and Script Libraries

In addition to these, the two Domino servers that you use should be configured in the same domain, sharing the same Domino Directory. You should also have the QuickPlace server set for Domino Directory integration. However, QuickPlace users do not need to be registered in the Domino Directory, thereby allowing QuickPlace Managers to create new users or pick them from the directory as necessary.

10.3.2 Building the example

The remainder of this section describes the steps that were used to build the example for Millennia using QuickPlace, Domino.Doc, and the Domino Designer. These steps build a complete application that allows a QuickPlace user to quickly designate a page to be added to the Domino.Doc library without the need to access the Domino.Doc library directly.

The steps in the example are described using Domino Designer R5.0. Users of Designer for Domino R4.6 will sometimes experience slightly differently wording or a different window layout in the Designer.

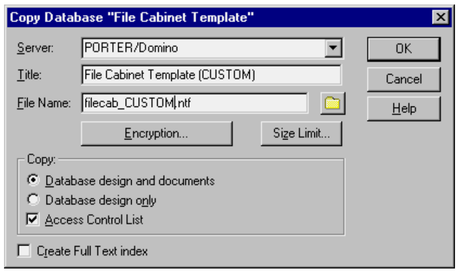
We only describe the parts of the Domino design elements that are relevant to the example. We assume some knowledge of Domino programming concepts such as forms, views, agents and so on. If you want to learn more about Domino programming, refer to the redbook *Lotus Domino R5.0: A Developer's Handbook*, IBM form number SG24-5331, Lotus part number CT6HPIE.

10.3.2.1 Setting up the Domino.Doc server

You need a Domino.Doc 3.0 installation to see the results of this example, and the user ID used for the modifications must have Domino.Doc Administrator access.

While this section does not list all the steps necessary to set up the Domino.Doc server, we show you what additional setup is required for the example. If you are making a test installation of Domino.Doc, you can install it using all the default values, and the user ID you are using will automatically get Domino.Doc Administrator access. For the library and file cabinet setup the information you need to follow the example is provided here.

Before continuing with the setup procedure of the Domino.Doc server, it is a good idea to make a database copy of the filecab.ntf using, filecab_QP.ntf as its filename. To do this, open filecab.ntf locally using the Domino Designer and select **File -> Database -> New Copy...**, giving it the new filename ensuring that you make this copy in the Data directory of the Domino.Doc server and not on your local machine. We will build our custom Document Type subform later.



Copy Database "File Cabinet Template"

Server: PORTER/Domino

Title: File Cabinet Template (CUSTOM)

File Name: filecab_CUSTOM.ntf

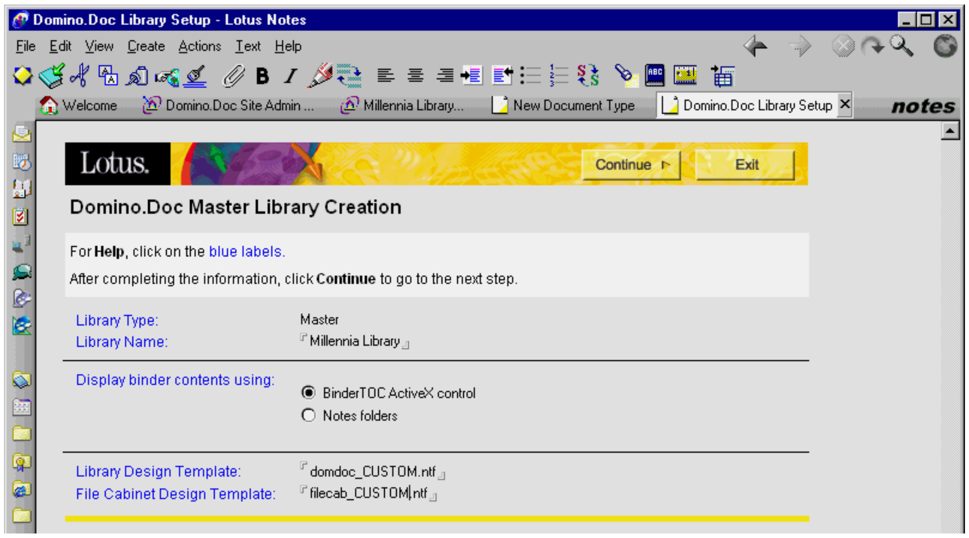
Copy:
- (●) Database design and documents
- ( ) Database design only
- [✓] Access Control List

[ ] Create Full Text index

Figure 102. Creating a New Copy on the Domino.Doc File Cabinet Template

Now you can create your library from the Domino.Doc Site Admin database. You should use Advanced Setup when you create your library so that you can give your library its proper name of Millennia Library.

You will also want to set filecab_CUSTOM.ntf as the default File Cabinet template and domdoc_CUSTOM.ntf for your library. You can use the default values for the rest of the choices under Advanced Setup.


Domino.Doc Library Setup - Lotus Notes
File Edit View Create Actions Text Help
Welcome
Domino.Doc Site Admin ...
Millennia Library...
New Document Type
Domino.Doc Library Setup
notes
Lotus.
Continue
Exit

Domino.Doc Master Library Creation

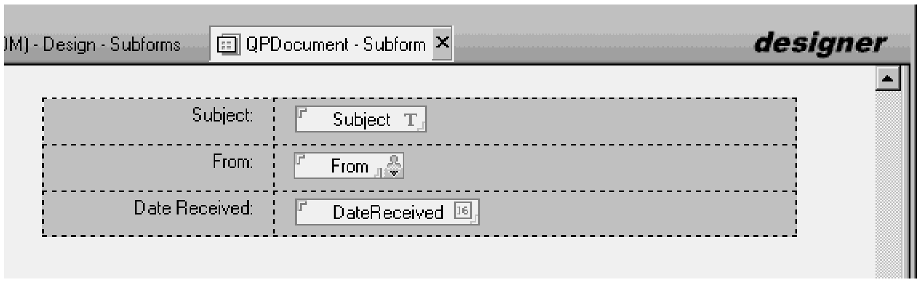
For **Help**, click on the blue labels.

After completing the information, click **Continue** to go to the next step.

Library Type:	Master
Library Name:	Millennia Library
Display binder contents using:	(●) BinderTOC ActiveX control ( ) Notes folders
Library Design Template:	domdoc_CUSTOM.ntf
File Cabinet Design Template:	filecab_CUSTOM.ntf

Figure 103. Domino.Doc advanced setup screen

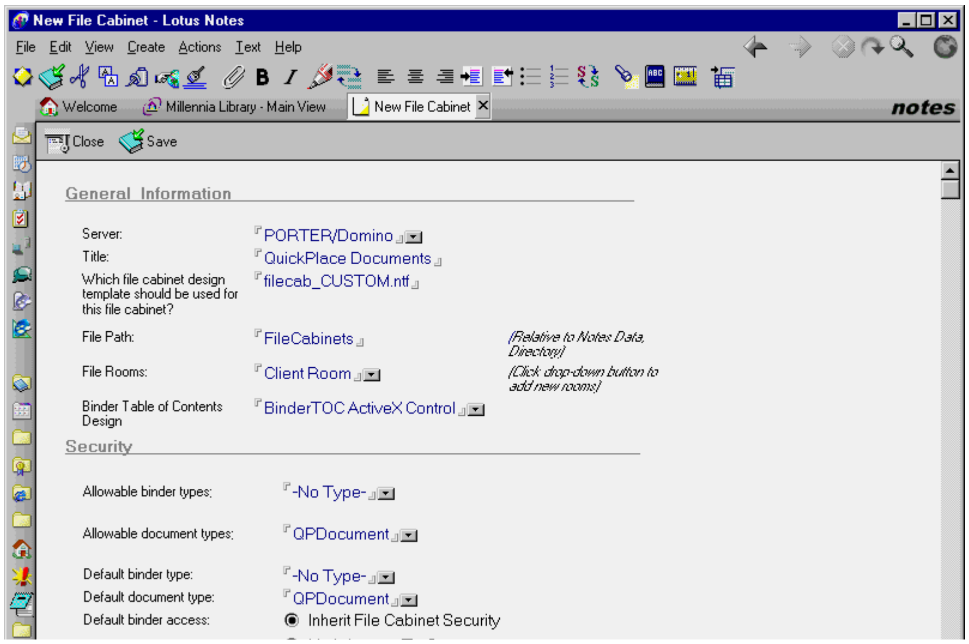
Once you have created your library, you must restart the Domino server and then open your newly created library to create a new file cabinet. But first, let us create a custom Document Type subform to be used for uploaded QuickPlace page content documents. Table 104 shows the subform that we need to create. Be sure to name it QPDocument when you save this new subform.



Subject:	Subject
From:	From
Date Received:	DateReceived

Figure 104. New Domino.Doc Document Type subform

Next create a file cabinet and name it QuickPlace Documents. You can create it using the default settings for the Binder Type, but the allowable and default Document Type fields should be set to the QPDocument Type that we created previously.


New File Cabinet - Lotus Notes
File Edit View Create Actions Text Help
Welcome
Millennia Library - Main View
New File Cabinet
notes
Close
Save

General Information

Server: PORTER/Domino
Title: QuickPlace Documents
Which file cabinet design template should be used for this file cabinet? filecab_CUSTOM.ntf
File Path: FileCabinets (Relative to Notes Data Directory)
File Rooms: Client Room (Click drop-down button to add new rooms)
Binder Table of Contents Design BinderTOC ActiveX Control

Security

Allowable binder types: -No Type-
Allowable document types: QPDocument
Default binder type: -No Type-
Default document type: QPDocument
Default binder access: ● Inherit File Cabinet Security

Figure 105. New Domino.Doc File Cabinet settings

10.3.2.2 Creating the QPRouter Processor database

Using the Domino Designer, create a new blank (no template) database on the Domino.Doc server in the server's Data directory. Create two views named *Inbox* and *Uploaded* with their design properties as listed in the following tables.

View "Inbox"

Design element	Property setting
View Selection	SELECT @IsUnavailable(ISDOMINODOC)
Column 1 Value	h_Name
Column 1 Title	Subject

View "Uploaded"

Design element	Property setting
View Selection	SELECT @IsAvailable(ISDOMINODOC)
Column 1 Value	h_Name

Design element	Property setting
Column 1 Title	Subject

Set this database to be a Mail-In Database called *QuickPlace Router*.

Now we need to create the Script Library that will be used to make calls to the Domino.Doc API. We will call this Script Library DDocLibrary.

10.3.2.3 DDocLibrary description

DDocLibrary is a LotusScript Class that defines the DDocArchiver object which performs the actual upload of the QuickPlace page content into a specific binder of a Domino.Doc library. In this section, we describe each method from a functional perspective.The entire class can be found on the IBM Redbooks Web site.

New(cabinetname, bindername)
This method instantiates the DDocArchiver object.

Archive(qpnote, bDeleteContent)
This method uploads the QuickPlace page to the Domino.Doc library.

CreateNewDocument(doctype, title)
This method creates a new Domino.Doc document in the library for the QuickPlace page contents. It is called by the Archive method.

CopyContentToDoc(qpnote, ddoc)
This method copies the QuickPlace page content into the new Domino.Doc document. It is called by the Archive method.

SaveAndCheckIn(versiontype, action, comment)
This method saves the new Domino.Doc document with the QuickPlace page content and performs a CheckIn of the new document as the specified version type.

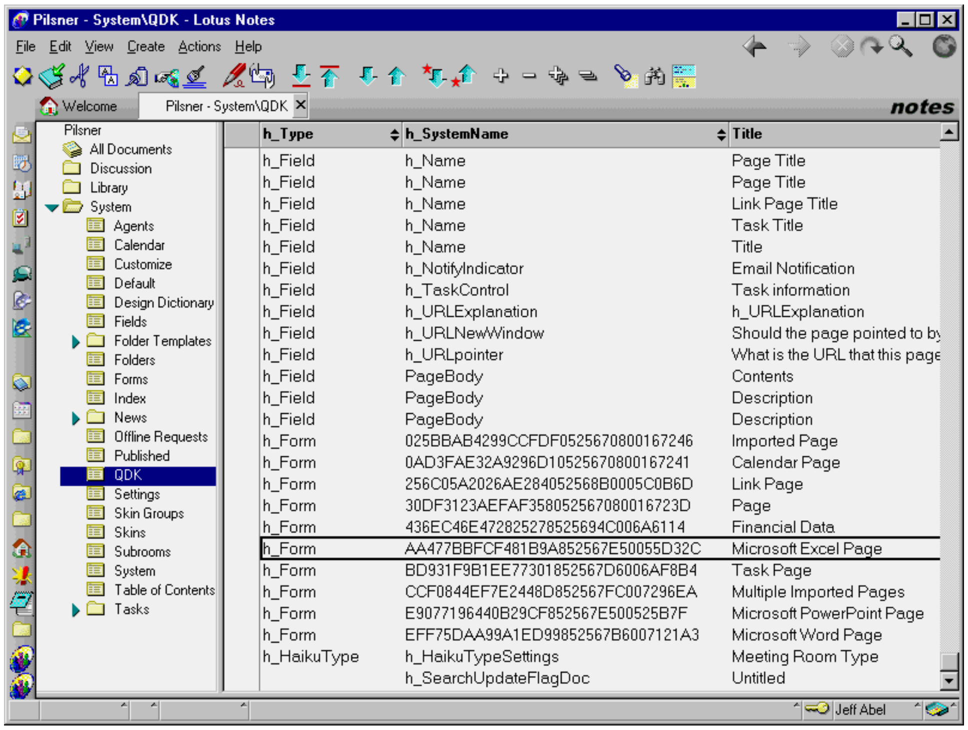
SetContents(filename)
This method sets the content of the new Domino.Doc document to a file attachment to handle the Word, Excel, or PowerPoint type of QuickPlace page content.

SetField(fieldname, fieldvalue)
This method sets the specified Domino.Doc profile field for the document type fields designed in the Document Type subform.

Now we need to create the scheduled agent that will manage the QPRouter Processor Inbox. We will call this agent QP-DDoc Upload and it will run every hour to check for new upload requests. You can adjust this schedule to fit with your organization’s needs.

10.3.2.4 QP-DDoc Upload agent described

This agent starts by setting four constants which represent the h_Form property of four QuickPlace form objects. These properties do not change from one QuickPlace to another. You can add to this list to handle other QuickPlace form objects such as the Task form or even your custom QuickPlace forms. Simply discover the h_Form property value for the form you wish to add a constant for by looking at the form object in the QDK view of the QuickPlace that you created the form in. Figure 106 shows what to look for.



h_Type	h_SystemName	Title
h_Field	h_Name	Page Title
h_Field	h_Name	Page Title
h_Field	h_Name	Link Page Title
h_Field	h_Name	Task Title
h_Field	h_Name	Title
h_Field	h_NotifyIndicator	Email Notification
h_Field	h_TaskControl	Task information
h_Field	h_URLExplanation	h_URLExplanation
h_Field	h_URLNewWindow	Should the page pointed to by
h_Field	h_URLpointer	What is the URL that this page
h_Field	PageBody	Contents
h_Field	PageBody	Description
h_Field	PageBody	Description
h_Form	025BBAB4299CCFDF0525670800167246	Imported Page
h_Form	0AD3FAE32A9296D10525670800167241	Calendar Page
h_Form	256C05A2026AE284052568B0005C0B6D	Link Page
h_Form	30DF3123AEFAF358052567080016723D	Page
h_Form	436EC46E472825278525694C006A6114	Financial Data
h_Form	AA477BBFCF481B9A852567E50055D32C	Microsoft Excel Page
h_Form	BD931F9B1EE77301852567D6006AF8B4	Task Page
h_Form	CCF0844EF7E2448D852567FC007296EA	Multiple Imported Pages
h_Form	E9077196440B29CF852567E500525B7F	Microsoft PowerPoint Page
h_Form	EFF75DAA99A1ED99852567B6007121A3	Microsoft Word Page
h_HaikuType	h_HaikuTypeSettings	Meeting Room Type
	h_SearchUpdateFlagDoc	Untitled

Figure 106. Using QDK view to find h_Form property value form QuickPlace form objects

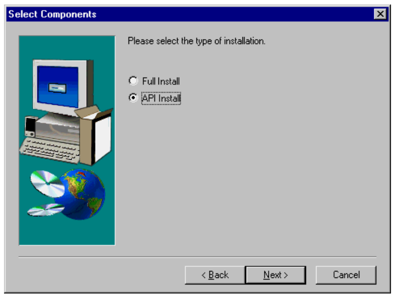
The agent then collects all new upload requests in the Inbox and for each determines which page type and calls the appropriate sub, either UploadAttachment(doc) for Word, Excel, and PowerPoint, or UploadDocument(doc) for QuickPlace standard pages.

This portion of the agent script looks as follows:

```
Sub Initialize
       Dim session As New NotesSession
       Dim db As NotesDatabase
       Dim dt As New NotesDateTime( "01/01/2000" )
       Dim dc As NotesDocumentCollection
       Dim doc As NotesDocument
       Dim i As Integer

       ' Set QuickPlace h_Form Constants
       Const qp2Page = "30DF3123AEFAF358052567080016723D"
       Const docPage = "EFF75DAA99A1ED99852567B6007121A3"
       Const xlsPage = "AA477BBFCF481B9A852567E50055D32C"
       Const pptPage = "E9077196440B29CF852567E500525B7F"

       ' Get all new documents for uploading
       Set db = session.CurrentDatabase
       Set dc = db.Search( "@IsUnavailable(ISDOMINODOC)", dt, 0)
       For i = 1 To dc.Count
           Set doc = dc.GetNthDocument( i )

           ' Determine which type of QuickPlace document we have and call
           ' the appropriate sub
           Select Case doc.h_Form(0)
           Case docPage, xlsPage, pptPage
               Call UploadAttachment(doc)
           Case qp2Page
               Call UploadDocument(doc)
           End Select

       Next
   End Sub
```

10.3.2.5 Necessary additions to the Domino.Doc server

In order for the QPRouter Processor database to upload the QuickPlace page content into Domino.Doc, you must have the Domino.Doc API and the Notes Client installed on the server machine. The Domino.Doc API we used for this example is part of the Domino.Doc Desktop Enabler install. When installing the Desktop Enabler, select Custom install, then you will only need to select the “API Install” option as shown in Figure 107.


Select Components

Please select the type of installation.

Full Install [✓ API Install]

Figure 107. API Install during Custom install of the Domino.Doc Desktop Enabler

When installing the Notes Client, you must use Release 5.0.3 concurrent with the requirements for Domino.Doc 3.0. You will only need to select the minimum options.

10.3.2.6 Setting up the QuickPlace server

Install the QuickPlace server to the existing Domino server and create a basic QuickPlace to be used for the example. Next create a new folder of type List called Domino.Doc for use as the vehicle for initiating the storage process. Modify the folder options for this folder to hide the New option to enforce the creation of QuickPlace content in folders other than this one. Next, create the QPRouter PlaceBot.

Creating the QPRouter PlaceBot

The QPRouter PlaceBot runs on a schedule and acts only on pages in the Domino.Doc folder. When it finds new pages in this folder, it mails each page and all of the page contents to the QPRouter Processor database on the Domino.Doc server and marks each as complete.

Now we create a new PlaceBot, call it QPRouter and add the following PlaceBot script:

```
Option Public

Sub Initialize
' This agent moves pages placed in the Domino.Doc folder to the QPRouter Processor
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim doc As NotesDocument
    Dim i As Integer
    Set db = session.CurrentDatabase
    Set collection = db.UnprocessedDocuments
    For i = 1 To collection.Count
        Set doc = collection.GetNthDocument( i )
        If Not doc.HasItem("SENT2DDOC") Then
            Call doc.Send( False, "QuickPlace Router@Domino" )
            doc.SENT2DDOC = "1"
            Call doc.Save ( True, True )
        End If
    Next
End Sub
```

That’s it--we now have an archiving solution where QuickPlace users can choose to have documents archived in Domino.Doc simply by moving them to a dedicated folder (we called ours Domino.Doc) in their Place.

Now let’s look at how QuickPlace can be integrated with Domino Workflow.

10.4 Integrating with Domino Workflow

QuickPlace has built-in workflow options designed to be configured and implemented by end users with little or no training in process management (See 6.1, “Defining a type of workflow” on page 135). End users can support their work processes and work practices without having to go through the lengthy setup procedures required to use an enterprise workflow platform.

However, at some point you may want to model more complex workflow or take advantage of process management offered by an enterprise workflow platform. If you wish to use conditional routing or automated activities, then you will need to integrate QuickPlace into a more robust workflow platform, or write your own workflow, using PlaceBots.

In this section we discuss integration with Domino Workflow, the enterprise workflow platform offered by Lotus. Just like QuickPlace, Domino Workflow applications use Domino databases and therefore offer some integration possibilities that are easy to discuss. Keep in mind that you could apply some of these methods to integrating another enterprise workflow platforms with QuickPlace.

Terminology

To describe the different approaches for integrating Domino Workflow, you need to be familiar with its terminology, as listed in the following table.

Term	Definition
Domino Workflow	A Domino-based workflow tool offered by Lotus. Domino Workflow has three primary components: a visual design tool called The Architect; a Domino application that performs workflow assignment and routing called The Engine’ and a visual end-user tool called The Viewer. Domino Workflow allows developers to quickly and easily incorporate sophisticated workflow in their Domino applications.
Process	The systematic definition of a set of activities that are necessary for completing a job or other kind of work.
Job	An instance of a process.

Term	Definition
Workflow Binder	A set of Domino documents that are routed through a workflow job.
Activity	A unit of work representing one of the steps in a process, or one of the steps required to work on a job.

Although we will discuss points of where the two platforms can integrate, you may need to consult product documentation or the redbook *Using Domino Workflow*, SG24-5963 to become familiar enough with Domino Workflow to make an application work.

10.4.1 Two approaches - depending on what's being wrapped

Although the flexibility inherent in the design of these two platforms supports a wide variety of integration possibilities, there are two basic scenarios for integration:

1. A user interacting with a Domino Workflow application spawns a Quickplace to support collaboration around a workflow or workflow activity.
2. A document created in a Quickplace initiates a workflow in a Domino Workflow application.

10.4.1.1 First scenario—Workflow wraps QuickPlace

In the first scenario, QuickPlace is entirely encapsulated in the Domino Workflow process. The life cycle of a Place is only as long as the Domino Workflow job that created it. Using this approach, Domino Workflow negotiates a short-term lease in transient housing; activity owners can meet with other participants to exchange information, make decisions, or take action that would otherwise have to be done offline.

A Domino Workflow application could use conditional logic or automated activities to cause QuickPlace to instantiate a Place and delete it when the job is finished.

As illustrated in Figure 108 on page 301:

1. An automated activity in Domino Workflow spawns a Place by sending a URL to the QuickPlace Server. The URL contains a unique name for the Place, which can be derived from the unique job identifier.
2. Within the Place, people who are both internal and external to the workflow collaborate.

3. When the collaborative activity is finished, Domino Workflow passes a URL to the QuickPlace server to delete the Place.
4. The workflow job is closed.

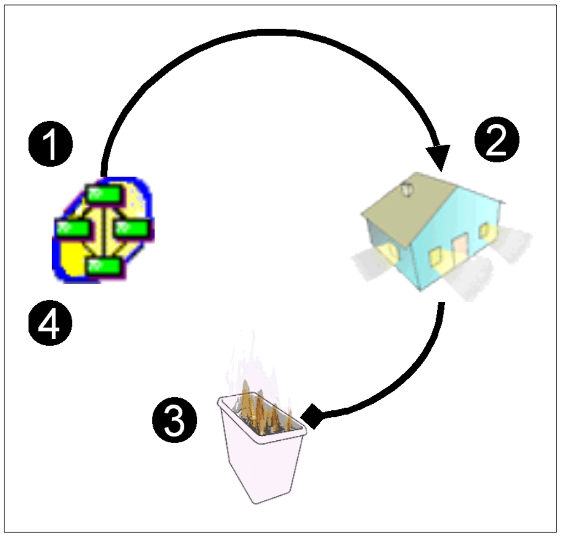


Figure 108. Workflow wraps QuickPlace

10.4.1.2 Second scenario—QuickPlace wraps Workflow

In this scenario, it is the Domino Workflow process that is encapsulated in QuickPlace. A document created in QuickPlace triggers the instantiation of a Domino Workflow job. This approach to integration is the same as integrating with any other Domino database—Domino Workflow applications *are* standard Domino databases.

In this scenario, you would use Domino Workflow to automate a process that is too complex to automate with the built-in workflow options in QuickPlace. Examples include enhancing the native QuickPlace document review cycle or creating application-specific workflow based on any type of Form available in QuickPlace.

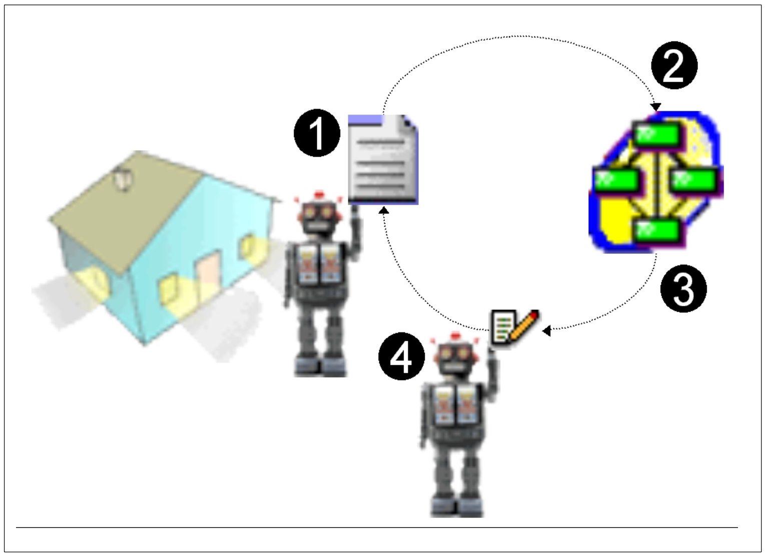


Figure 109. QuickPlace wraps Domino Workflow

As shown in Figure 109:

1. A PlaceBot in QuickPlace detects a document in a specified folder or created with a specified Form. The PlaceBot attaches Domino Workflow reserved fields to the document, identifying the document to tell Domino Workflow how to handle it as a workflow job. The PlaceBot sends the document to a Domino Workflow application database.
2. When the document from QuickPlace arrives in the Domino Workflow application database, Domino Workflow recognizes the reserved fields, triggering a workflow job.
3. During the course of the workflow job, automated activities in the Domino Workflow application database send updates of job status back to QuickPlace.
4. A PlaceBot, acting as a mail room attendant, associates the status update as a response to the original document that triggered the Domino Workflow job.

As a simple alternative to Steps 3 and 4, a link document to a view in the Domino Workflow application database provides updated status on *all* jobs.

10.4.2 Creating Places from the Domino Workflow environment

Creating a Place from a Domino Workflow application, or from any other application for that matter, is as simple as sending an appropriately formatted URL to the QuickPlace server. Within the context of a Domino Workflow application, the Place can be created either by user interaction via the Domino Workflow user interface, or automatically by the workflow engine. For information about how to create Places by sending a URL to a QuickPlace server, see 8.2.17, "URL to create a Place" on page 244.

10.4.2.1 Places from Domino Workflow or Notes user interface

Using this method, an interface element such as an action button is included in one of the workflow binder documents. This button calls an agent that, using LotusScript, sends the creation URL to the Quickplace server. The Domino Workflow application developer then provides the user simple navigation to the newly created Place from the Domino Workflow binder, as the name of the Place was a required parameter in the URL sent to the QuickPlace server.

Example: a script class to instantiate a QuickPlace

This class allows you to pass a URL to the constructor that can be sent to a QP server to generate a QP. You can also simply set the required Properties and call createQuickplace. The IsValid property checks to make sure the required properties have values. You can modify the following code to use in your own Notes or Domino Workflow environment (the code refers to a function which is listed in Appendix I, "ReplaceSubstring_ function" on page 399):

```
Class ccQuickplaceCreator

   Private m_st As String
   Private m_stProtocol As String
   Private m_stServerName As String
   Private m_stPlaceType As String
   Private m_stPlaceName As String
   Private m_stUserName As String
   Private m_stPassword As String
   Private m_stEmailAddress As String
   Private m_stReturnURL As String

   Property Set Protocol As String
      m_stProtocol = Protocol
   End Property

   Property Get Protocol As String
      If m_stProtocol = "" Then
         m_stProtocol = "http"
      End If
      Protocol = m_stProtocol
   End Property

   Property Set PlaceType As String
      m_stPlaceType = PlaceType
   End Property

   Property Get PlaceType As String
      PlaceType = m_stPlaceType
   End Property

   Property Set PlaceName As String
      On Error Resume Next
      Call ReplaceSubstring_( PlaceName, " ", "+" )
      m_stPlaceName = PlaceName
   End Property

   Property Get PlaceName As String
      PlaceName = m_stPlaceName
   End Property

   Property Set UserName As String
      m_stUserName = UserName
   End Property

   Property Get UserName As String
      UserName = m_stUserName
   End Property

   Property Set Password As String
      m_stPassword = Password
   End Property

   Property Get Password As String
      Password = m_stPassword
   End Property

   Property Set Emailaddress As String
      m_stEmailaddress = Emailaddress
   End Property

   Property Get Emailaddress As String
      Emailaddress = m_stEmailaddress
   End Property
```

```
   Property Get DefaultReturnURL As String
      DefaultReturnURL = "http://" + Me.ServerName + "/Quickplace/" + 
Me.PlaceName + "/main.nsf?OpenDatabase"
   End Property

   Property Set ReturnURL As String
      m_stReturnURL = ReturnURL
   End Property

   Property Get ReturnURL As String
      If m_stReturnURL = "" Then
         m_stReturnURL = Me.DefaultReturnURL
      End If
      ReturnURL = m_stReturnURL
   End Property

   Property Set ServerName As String
      m_stServerName = ServerName
   End Property

   Property Get ServerName As String
      ServerName = m_stServerName
   End Property

   Property Get ConstantParmString As String
      ConstantParmString = 
"/QuickPlace/QuickPlace/CreateHaiku.nsf?OpenDatabase&PresetFields=h_SetEdi
tCurrentScene;h_CreateManager," + _
      "h_EditAction;h_Next,h_SetCommand;h_CreateOffice"
   End Property

   Property Get CreateURL As String
      On Error Resume Next
      Dim st$
      If m_st = "" Then
         '--- if no url passed then, generate one
         st = Me.Protocol + "://" + Me.ServerName + Me.ConstantParmString + 
_
         ",h_PlaceTypeName;" + Me.PlaceType + _
         ",h_Name;" + Me.PlaceName + _
         ",h_UserName;" + Me.UserName + _
         ",h_SetPassword;" + Me.Password + _
         ",h_EmailAddress;" + Me.Emailaddress + _
         ",h_SetReturnUrl;" + Me.ReturnURL
      Else
         st$ = m_st
      End If
      Call ReplaceSubstring_( st, " ", "+" )
      CreateURL = st
   End Property

   Property Get IsValid As Integer
      If Me.PlaceName = "" Then
         Exit Property
      End If
      If Me.UserName = "" Then
         Exit Property
      End If
      If Me.Password = "" Then
         Exit Property
      End If
      IsValid = True
   End Property

   Sub new( arg_varURL )
      On Error Resume Next
      If Not Isempty( arg_varURL ) Then
         m_st = arg_varURL
      End If
   End Sub

   Function CreateQuickplace() As Integer
      Dim db As notesdatabase
      Dim ns As New notessession
      Set db = ns.CurrentDatabase
      Call db.GetDocumentByURL( Me.CreateURL )
      CreateQuickplace = True
   End Function

End Class
```

10.4.2.2 Using an automated activity to create Places

Domino Workflow models workflow as a series of activities that are performed to complete a job. The Architect supports creation of activities that require no human intervention, called *Automated Activities*. Automated Activities may be configured to send mail, execute a LotusScript agent, or call a server program.

The simplest way to create a Place automatically within a workflow is to configure an Automated Activity to execute an agent that, via LotusScript, sends the appropriate URL to the QuickPlace server. This Automated Activity could be executed within every workflow job, or could be executed conditionally, based on routing relations defined in The Architect.

10.4.2.3 Initiating a job in Domino Workflow

Domino Workflow provides three mechanisms for initiating a workflow job:

1. The user interactively creates a document in the Domino Workflow application interface.
2. Mail is sent to a Domino Workflow application database configured as a mail-in database. Settings in the Domino Workflow application setup document determine how the mail fields are mapped to fields the engine needs to initiate a job. Jobs can be started when a document with designated key fields is mailed to or created in the application database.
3. A back-end document containing a series of reserved items is created in the Domino Workflow application database. The items specify which process to initiate, and other information for the identification and handling of the job.

10.4.3 Initiating a Domino Workflow job from QuickPlace

Regardless of the scenario integrating Domino Workflow, you must consider several factors when designing this type of integration:

1. Initiating a workflow job in the Domino Workflow application based on a the creation of a Quickplace page
2. Maintaining a link between the original QuickPlace page and the binder created in the Domino Workflow application so that the user may easily navigate between them
3. Optionally updating QuickPlace to reflect the status of the workflow job

10.4.3.1 Form-based initiation of Domino Workflow jobs

Form-based initiation is practical when the user who creates or sends this document into the application database is aware of the process that will be initiated. The process, job name, and priority can be entered directly. However, if the process name is built into the form, simply creating a document based on the Form will begin the job.

Follow these steps to create the Form in an existing Domino Workflow application database:

1. Open the application database.
2. Choose **View - Design** from the menu. A design navigator appears.
3. Click the Forms view. In the right panel, open (OS Form-based Initiation Template).

4. Highlight and copy the five fields located in this template; then close the template.
5. Choose **Create - Design - Form** to create a new form.
6. Paste the fields into the new form. See Table 47 for further information regarding the fields you paste into the new Form.
7. Add your own Form design and save the Form. You can view the Form in a browser, save as HTML, and import it into QuickPlace as described in Appendix 6.4, “Upload a manually created HTML Form” on page 146.

10.4.3.2 Initiating a Domino Workflow job with a custom Form

Any Form containing the fields in the Form (OS Form-based Initiation Template) can be used to start a job (see Table 47). The Domino Workflow Backgrounder agent searches documents in the application database. If it finds the field InitiateOS in a document, and if the field is not empty, the document will be used to start a job.

10.4.3.3 Initiating a Domino Workflow job through polling

This requires a special tool in the Domino Workflow application to check for the presence of documents in QuickPlace that contain the Domino Workflow reserved fields.

Table 47 lists the fields that are used in form-based initiation of Domino Workflow. Note that formulas cannot be use in any of these fields.

Table 47. Fields used in form-based initiation of Domino Workflow

Field name	Purpose
NewProcessNameOS	Determines which process should be started by the form.
NewJobNameOS	Determines the name of the job started by the form.
NewJobPriorityOS	Determines the job priority.
MailStatusOS	If the field is empty ("") or contains "2", the document becomes the main document in the binder. If the field contains "3", the document will be deleted. If the field has any other content, the document becomes a binder document.

Field name	Purpose
InitiateOS	If the field is empty, or contains "no", the document won't be used to start a job. If the field contains "yes", the document will start a job based on the other fields in this table.
EnternalInitiatorOS	You may optionally use this field to specify the person's name that is responsible for initiating this new job. The value of this field is transferred to the cover document during initiation. Within the process design, you can refer to this field using the Job Property "External Initiator". We recommend that you use a canonical user name.

10.4.3.4 Initiate the Domino Workflow job using a PlaceBot

The most important thing to remember is that regardless of how it gets there, a document that initiates a job in the Domino Workflow application database must have all of the reserved fields described in Table 47. Your PlaceBot will send a document from QuickPlace to the Domino Workflow Application database, which must be configured as a mail-in database.

However, you must modify the document prior to sending. Start with the LotusScript PlaceBot in the QPRouter example described in 10.3.2.6, “Setting up the QuickPlace server” on page 298 and modify it to do the following:

1. Change h_Form to "256C05A2026AE284052568B0005C0B6D". (This is the unique ID of the page that represents the link form; when we see this, we redirect to the URL supplied. Save the original value of h_Form so you can change this back once the workflow is complete, so it can be read in QuickPlace.)
2. Set h_URLpointer to the URL you want to redirect to.
3. Set h_URLNewWindow to “yes” or “no”, to indicate whether you want to open your version of the page in a new window.
4. Set InitiateOS to “yes”.
5. Set ExternalInitiatorOS to the computed username of the author of the document, preferably in canonical format.
6. Set NewJobNameOS to compute the value of h_Subject.

7. Send the document to the Domino Workflow application database using one of the following methods:
   a. LotusScript Sendmail Method. This is the preferred method, but will only work if QuickPlace is in the same Domino domain as Domino Workflow.

      The syntax of the LotusScript method is:

      Call notesDocument.Send(attachForm [, recipients])

      To learn more, look up the method in the Domino Designer online help.
   b. SMTP (standard Internet-type mail). When you cannot send and receive Notes mail to the Domino Workflow application database, you may have to use this method.

      You will have to push the values of all of the fields listed in Steps 1 to 6 through the Subject field (h_Subject in a QuickPlace document). In the Domino Workflow application database. you will need to parse the field names and values out of Subject and back into the reserved field names.

10.4.3.5 Getting the binder information back to QuickPlace

If you are satisfied with letting Domino Workflow handle the remainder of the process, then creating a way to check the status of jobs is as easy as creating a Link Document in QuickPlace to point to a view in the Domino Workflow application database.

The simplest way for Domino Workflow to update QuickPlace about the status of a job is to send mail to the Place. A PlaceBot such as the Mail Room Attendant (see our example application in 7.5, “The QuickPlace Mail Room Attendant” on page 186) can be configured to recognize the status message from Domino Workflow to route the status to the appropriate folder. You might want to create a fully customized application to associate the status information coming from Domino Workflow with the document that started the process.

This concludes our discussion of QuickPlace integration with Domino Workflow. We now move on to discussing the integration of QuickPlace in a portal.

10.5 Integrating with a portal

The power and flexibility of QuickPlace make it a dynamic part of your corporate portal strategy. QuickPlace offers easy integration into any Web portal. You can customize QuickPlace make this integration seamless.

A *portal* is a metaphorical door. A Web portal, by definition, encapsulates other Web content. A *corporate portal* is a single point of personalized access for the pooling, interaction, and distribution of organizational knowledge. While QuickPlace can integrate into many types of portals, in this redbook we only discuss the corporate portal, as it is closely aligned with the project team focus of QuickPlace.

We briefly introduce the different types of portals, in order to set the scene for the kind of QuickPlace portal integration we will discuss.

10.5.1 Extending the portal metaphor without straining it

People need a virtual place to share things. QuickPlace serves people, places, and things by linking them together so that people throughout an organization have virtual places in which they can interact and manage the things that they need to achieve their business tasks—all from a single point of access.

It is not overreaching to state that portals are the doors through which people pass to get to places. The difficulty lies in grasping the *many levels* of portals as they concurrently serve the individual person, the team, the community, the corporation, and the enterprise (see Figure 110 on page 312). At each level, a portal provides access to business objects—content and services, including other portals.

We once used terms like intranet and extranet to define the boundaries between inside and outside the organization. But just as the lines of business are blurring, so too the concept of the portal complicates the lines of access. Some layers are public; some are personal. All may have parts that are accessible or inaccessible to some members of the enterprise. Indeed, every Place can be a portal in its own right.

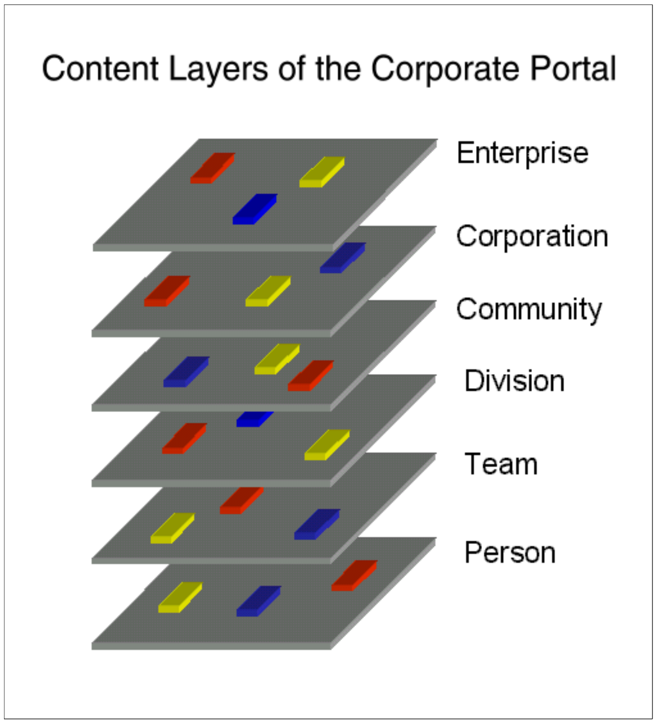


Figure 110. The many levels of portals

Note: QuickPlace is primarily geared to create *team portal*s. However, portals for the individual person, the team, the organizational hierarchy, and communities are all functionally similar. They exist concurrently to provide access to the same universe of content objects, but differ in their presentation and services offered.

Early metaphors for the Web and its immediate precursors were closely tied to print media. Electronic newsletters and early sites on the Web were presented and, more importantly, *developed* like newspapers. This way of thinking was very constraining. The power of hyperlinks, multimedia, and interactive content present many opportunities to move beyond the linear and static nature of print media, but only if we advance our thinking. We needed a better metaphor.

Enter the metaphor of *place*. We imagine the virtual place as being like a house with rooms. We can enter the house (place) if we have a key (password), and likewise can enter certain rooms in the house (the main and inner rooms in QuickPlace). While in these rooms, we can leave items for others to read sometime after we leave (asynchronous collaboration or discussion) or, while we are in the virtual place, we can discuss these items in real time (chat).

If we think of our team as a family, we can imagine assigning rooms for various members to look after, and we can imagine the sense of community that we develop by sharing space responsibly. We can also imagine how that community can break down when members of the family never visit, or don't clean up their rooms, or the head of household doesn't perform needed maintenance.

The place metaphor of a house with rooms is very powerful; however, it falls short in several ways. Role-based access control allows us to control not only who can go where, but what objects look like and what people can do with them, based on their role in the group or even more complex sets of events and conditions. The items which we see in one place can simultaneously be seen from an infinite number of places, reformatted to reflect the business context. What new metaphor will advance our thinking to the next level?

10.5.1.1 Integrating QuickPlace into a portal architecture

Without any customization at all, QuickPlace allows people to create portals optimally geared for the needs of a team. These team portals are islands of self-sufficiency, but as we have discussed, it's not long before you will want to move beyond independence to create synergy with other enterprise application platforms and establish relations with the other islands. If Places are free-standing structures, then, in the context of the corporate portals, community portals are neighborhoods, while other levels of portals may act like a hierarchy of territories or provinces.

It is not uncommon for people to work on several teams simultaneously and access a wide range of *applications*, *collaborative services* and *personal services*. Let's discuss these terms further:

Applications
These run on the top tier of the architecture. They include Personal Places and Community Places, which provide routes into content and shared workspaces.

Collaboration Services
These include real-time tools that support people awareness and instant messaging, as well as online team workspaces that provide structured content and learning tools. E-mail is also a component of the collaboration services tier.

Personal Services
These include desktop applications, the Web, and back-end systems.

Someone who is a member of more than one Place will naturally want a single place from which to navigate to Places, and to organize all of their other services. You can integrate Places created using QuickPlace to be launched from your Web portal.Table 48 lists the more common portal services and describes their relation to QuickPlace.

Table 48. Common portal services and how they are supported by QuickPlace

Common portal service	How QuickPlace supports it
Common look-and-feel, tailored to context	QuickPlace skins can be manipulated to create a seamless integration into the Portal look-and-feel. Themes can be selected to match the context of their use, depending on either the person accessing the Place or depending on whether the Place was accessed directly or through a portal (see 10.5.2, “The mechanics of simple portal integration” on page 315).
Security services	Through server settings alone, QuickPlace can be fully integrated into corporate directories, including any Domino directory, NT directories, and LDAP directory (see 3.3.3, “Directory integration” on page 38). Single sign-on through session-based authentication allows seamless access of multiple QuickPlace and portal resources (see 3.3.5, “Session-based authentication” on page 41).
Search and browse services	QuickPlace Places are Domino databases with full-text indexing capability. Any search service that can index a Domino database can search QuickPlace. However, training a standard search engine to recognizing QuickPlace Objects is more challenging. Two sample applications, if combined, could provide an integrated search and browse of a QuickPlace server. TheXray (see 7.6.1, “TheXRay server status reporter agent” on page 195) shows the location of all Places and Rooms within Places. Mapperizer (see 7.3, “LotusScript PlaceBots” on page 170) arranges objects within Rooms—Pages, Folders, and Members—in a hierarchical display.

Common portal service	How QuickPlace supports it
Diverse content integration	Through LotusScript or Java, QuickPlace can negotiate full access to legacy databases. Newsfeeds and other e-mail can be sent directly into Places and then automatically managed by PlaceBots (see 7.5.2.1, “Writing the MailRoomAtendant.lss PlaceBot” on page 193). Importing HTML and MS Office documents into QuickPlace allows them to be shared on the Web, while round-trip editing allows them to be edited, using the applications that created them (see Table 6.3).

Table 49 lists the core portal architecture that QuickPlace provides.

Table 49. Core portal architecture provided by QuickPlace

Core portal architecture	How QuickPlace supports it (with reference)
Informal or *ad hoc* collaboration	QuickPlace supports informal collaboration with colleagues across the organization, as Place Managers can invite others into their places, and end users can create Places using a URL or a front- end portal application that computes a URL to send to a QuickPlace server (see 10.2, “A Domino application to handle requests for new Places” on page 281).
Team collaboration	QuickPlace supports team collaboration that is either ad hoc, or in managed “virtual teams”. Teams may create secure shared space for pre-release or sensitive work.

10.5.2 The mechanics of simple portal integration

The fastest and simplest way to integrate one Web application with another is to create links from one to the other.

You can use hardcoded links or build links programmatically to:

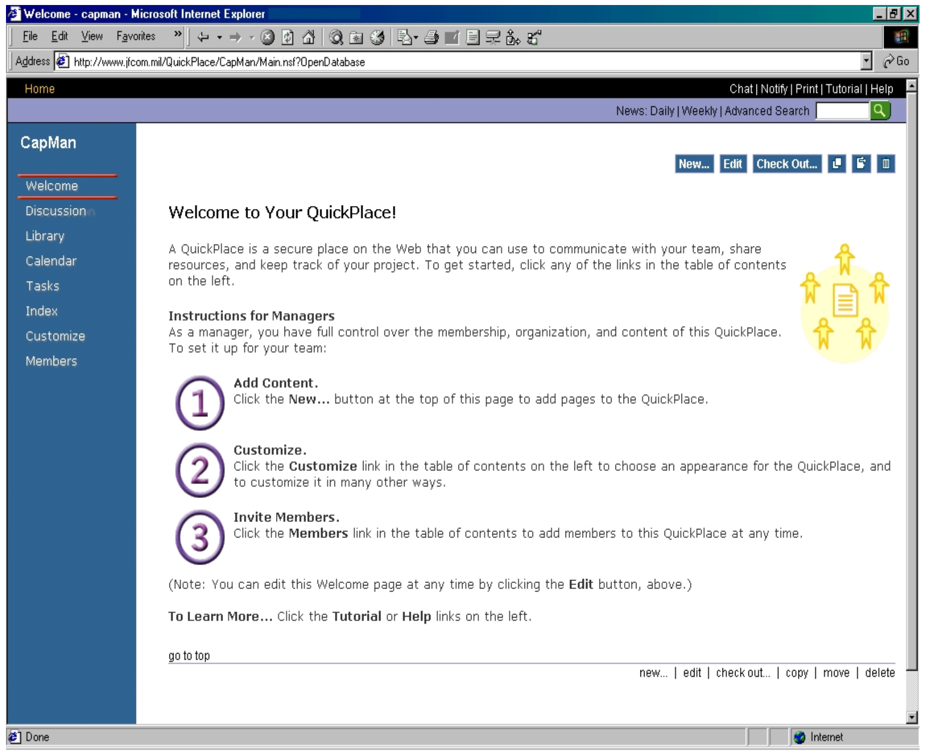
- Link to other places in frames or windows
- Link to objects in QuickPlace
- Send commands to the directly to QuickPlace server from the portal interface
  - Navigate to a Place, Room, View, Folder, Page or Form
  - Create a QuickPlace Object
  - Execute a search in a Place

Individuals and communities manage access to QuickPlace content of their selection. The portal is the common wrapper and organizing point for diverse content.

10.5.2.1 Presentation issues

While simple solutions are often most powerful, a little refinement to simple links can go a long way in achieving seamless integration. Specifically, you should pay attention to the sizing and navigation when integrating QuickPlace into a portal.

Millennia consulting would like to integrate QuickPlace into its corporate portal, the *M-PowerPortal.* It would be simple to just point a Web page frame to the Place. Using this method displays the QuickPlace in its entirety, including all navigation and tools. However, depending on the layout of your Place, this can mean a lot of duplication in navigation and probably a great deal of confusion for the user (see Figure 111).



Welcome - capman - Microsoft Internet Explorer
File Edit View Favorites
Address http://www.jfcom.mil/QuickPlace/CapMan/Main.nsf?OpenDatabase
Go
Home
Chat | Notify | Print | Tutorial | Help
News: Daily | Weekly | Advanced Search
CapMan
Welcome
Discussion
Library
Calendar
Tasks
Index
Customize
Members
New...
Edit
Check Out...
go to top
new... | edit | check out... | copy | move | delete
Done
Internet

Welcome to Your QuickPlace!

A QuickPlace is a secure place on the Web that you can use to communicate with your team, share resources, and keep track of your project. To get started, click any of the links in the table of contents on the left.

Instructions for Managers
As a manager, you have full control over the membership, organization, and content of this QuickPlace. To set it up for your team:

1. **Add Content.**
Click the **New...** button at the top of this page to add pages to the QuickPlace.

2. **Customize.**
Click the **Customize** link in the table of contents on the left to choose an appearance for the QuickPlace, and to customize it in many other ways.

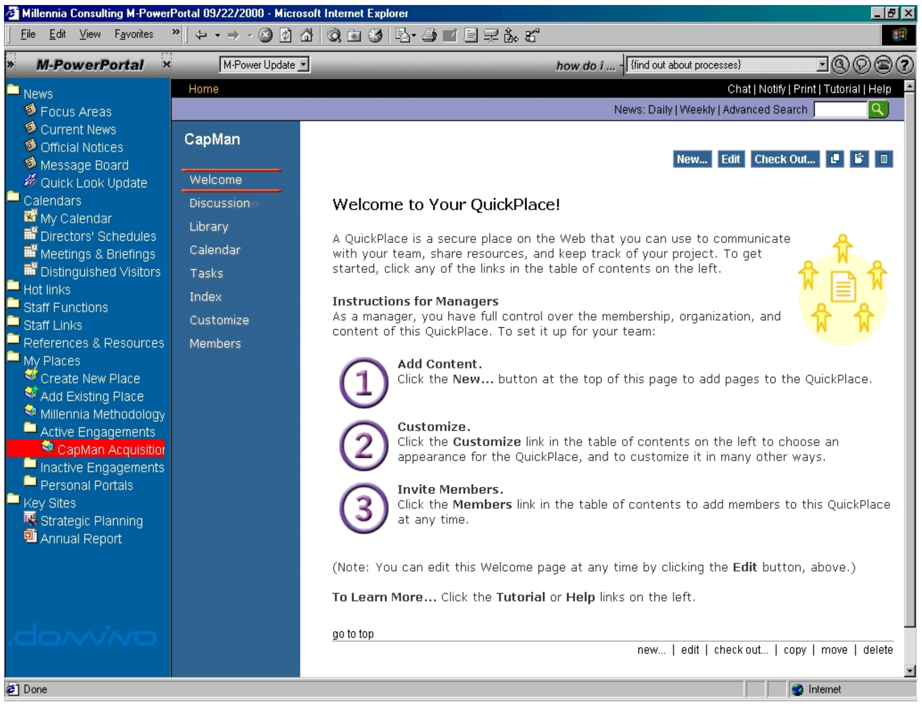
3. **Invite Members.**
Click the **Members** link in the table of contents to add members to this QuickPlace at any time.

(Note: You can edit this Welcome page at any time by clicking the **Edit** button, above.)

To Learn More... Click the **Tutorial** or **Help** links on the left.

Figure 111. How members see CapMan Acquisition Place from outside the M-PowerPortal

So we choose a standard Theme or create a custom Theme that would be displayed for portal users viewing the Place through a frame in the portal interface. This Theme, intended for use with the portal, still provides the user with a means to navigate through the Place. But it forces the user to use the portal interface to move around. By selecting what elements from QuickPlace to include in a Theme used from *inside* the portal, you guide the user into using the more robust tools of the portal, such as the common search or Sametime Awareness.



Welcome to Your QuickPlace!

A QuickPlace is a secure place on the Web that you can use to communicate with your team, share resources, and keep track of your project. To get started, click any of the links in the table of contents on the left.

Instructions for Managers
As a manager, you have full control over the membership, organization, and content of this QuickPlace. To set it up for your team:

1 **Add Content.**
Click the **New...** button at the top of this page to add pages to the QuickPlace.

2 **Customize.**
Click the **Customize** link in the table of contents on the left to choose an appearance for the QuickPlace, and to customize it in many other ways.

3 **Invite Members.**
Click the **Members** link in the table of contents to add members to this QuickPlace at any time.

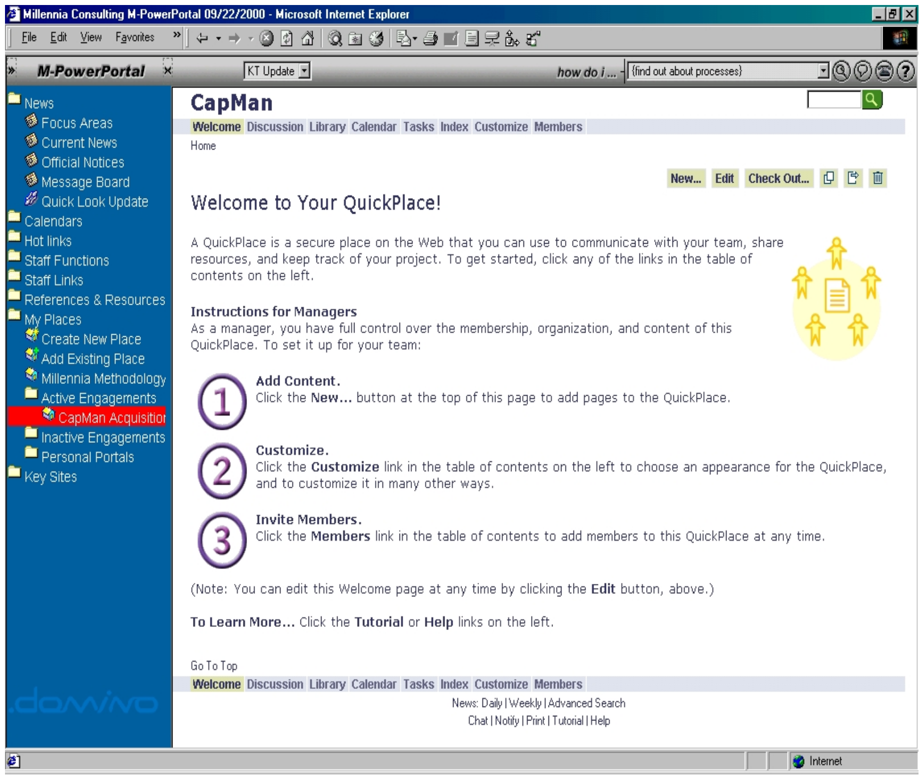
(Note: You can edit this Welcome page at any time by clicking the **Edit** button, above.)

To Learn More... Click the **Tutorial** or **Help** links on the left.

Figure 112. Standard CapMan Acquisition Place, standard Theme viewed through M-PowerPortal

Note: Figure 112 shows the standard CapMan Acquisition Place with a standard Theme, as viewed through the M-PowerPortal. As you can see, having two sets of vertically-oriented navigation is confusing.

To make a Place that will be wrapped with a portal interface using frames, you need to be conscious of the layout of the portal navigation and design your Themes accordingly. For example, the portal navigation depicted in the M-PowerPortal has a vertically-oriented bank of links on the left side of the interface. To avoid a confusing layout of navigation, use a Theme where the main navigation runs horizontally across the screen, such as the Banner Theme found in QuickPlace Standard Themes (see Figure 113 on page 318).



Welcome to Your QuickPlace!

A QuickPlace is a secure place on the Web that you can use to communicate with your team, share resources, and keep track of your project. To get started, click any of the links in the table of contents on the left.

Instructions for Managers
As a manager, you have full control over the membership, organization, and content of this QuickPlace. To set it up for your team:

1 **Add Content.**
Click the **New...** button at the top of this page to add pages to the QuickPlace.

2 **Customize.**
Click the **Customize** link in the table of contents on the left to choose an appearance for the QuickPlace, and to customize it in many other ways.

3 **Invite Members.**
Click the **Members** link in the table of contents to add members to this QuickPlace at any time.

(Note: You can edit this Welcome page at any time by clicking the **Edit** button, above.)

To Learn More... Click the **Tutorial** or **Help** links on the left.

Figure 113. How user views the same Place through a frame in the portal

10.5.2.2 Handling direct access and portal access

There are cases where users sometimes access a QuickPlace through their portal and some times go directly to it - or where a certain group uses a portal while another group (for example, external team members) must use direct access.

So how can you give both direct access users and portal users an optimal user interface experience? You could select a Theme that works well with a portal, and then simply hope that direct users also get good functionality from it. However, there is another way.

QuickPlace allows you to intercept the command that sets the Theme for a user through its event interface. This allows you to determine whether the user is a portal user or a direct user, and then select a Theme that is appropriate for that specific user.

There are several ways to determine what kind of a user you are dealing with. If you are distinguishing between different groups of users (for example, internal and external), they may reside in different groups in your user directory.

If you are distinguishing between direct users and users coming through another application like a portal, it's very likely that you can test for a cookie. Most Web application uses cookies, so you can test for the cookies you are interested in, and set the Theme accordingly. If no cookies are present, you can assume that you have a direct user.

This technique is used in QuickPlaces supplied with the Lotus K-station portal, where the user sees different Themes in direct and portal access.

See 9.2, "Using the QuickPlace event interface" on page 266 to get more information about how to work with the QuickPlace event interface. There is an event called h_GetSkinGroupName that you can use to set a Theme yourself. For similar examples, look at the DevCon 2000 samples in the QuickPlace DevZone at:

`http://www.quickplace.com/devzone`

This site contains an example where the Theme used is determined by which browser is being used, and another example tests to see whether the actual user has a preferred Theme to use and then sets that Theme up.

Integrating an application into a portal can be a huge topic when you start to discuss things like sing-sign-on, dynamic data transfer between portal and application, and so on. In this section we've only scratched the surface in discussing how, with a simple Theme selection, you can improve the portal user experience dramatically.

10.6 Summary

In this chapter we looked at how QuickPlace can integrate with other applications and system. We stayed mostly within the Lotus family, and discussed integration with a Notes Client, a Domino application, Domino.Doc, Domino Workflow, and finally, integration with a portal.

Chapter 11. Creating PlaceTypes and a Turnkey Server

In this last chapter we pull together many of the things covered in the previous chapters in our discussion of PlaceTypes and Turnkey Servers.

PlaceType support is one of the key strengths in QuickPlace. It gives the user, in cooperation with the QuickPlace administrator, an easy way to take a snapshot of those Places that apply best practices in the work process and application functionality and make them available for users that create new Places.

We use example scenarios in our descriptions in this chapter, so we start by briefly describing these scenarios (that is, scenarios beyond our *general* scenario described in 2.3, “Our redbook scenario - Millennia Consulting” on page 17).

Then we walk through what the user (the administrator of a single Place) and the administrator (for the whole QuickPlace server) need to do to promote a Place as a PlaceType and make it available for new users. Following that, we show how the user specifies a specific PlaceType during the creation of a Place.

We also walk through a description of a scenario where our sample company Millennia set up a Place that was based on a PlaceType.

In the second part of this chapter we go a step further and build a Turnkey Server where we can include our customized functionality and PlaceTypes, and make all of it available on a new server in a one-shot installation.

11.1 Use of PlaceType - two examples

In this section we use two different examples to illustrate our work with PlaceTypes and Turnkey Server. In one example, PlaceTypes are created to deploy applications within the same company using a Turnkey Server. In the other example, a company creates PlaceTypes based on its know-how, and sells its know-how packaged as a QuickPlace Turnkey Server.

11.1.1 Example: Millennia opens an office in Sydney, Australia

Millennia plans to open a new office in Sydney, Australia. They want to take all their customized software with them, including their customized Places. As new people are staffing this office, Millennia just needs the Places themselves, and they will people them with the new employees in Sydney.

To meet these requirements, Millennia needs to port their complete QuickPlace server. They create PlaceTypes from all existing Places, including all Placebots and Themes. The existing users are not included in the PlaceTypes, as new people are coming into the Sydney office. When the PlaceTypes have been set up, a Turnkey Server will be created that includes the Millennia PlaceTypes.

11.1.2 Example: Mudge & Mudge buys Turnkey Server from Millennia

You may remember Doug Mudge, legal counsel to CapMan Software. Doug is a partner in a small law firm (six attorneys and six support staff) in San Francisco that represents high-technology startups while they are still too small to hire a full-time attorney. Doug’s business is all about making his clients feel like he is an indispensable part of their tightly-knit teams.

Doug is really impressed by the way his participation in the CapMan Acquisition Place kept him closely connected to the team. After TheRock’s acquisition of CapMan was complete, Doug asks Millennia attorney Nancy Vargo about having his own QuickPlace server. Nancy refers Doug to a Millennia team of consultants and developers who can understand and fulfill Doug’s need. Millennia is familiar with all of the key processes of running a law firm. These processes are typically highly structured in a large law firm, while small-to-medium size firms operate with a few, broadly defined work areas, where flexibility and unstructured work practices often prevail.

The key process areas at Mudge & Mudge are *Staff Management, Billing, Office/Library Management,* and *Client Management*. Client Management is actually a conglomeration of several processes including litigation/case management, legal research, legal products, and calendar management—all centered around the client.

Millennia designs a PlaceType to support each of these key process areas. They have some PlaceTypes that are already very close to the needs of Mudge & Mudge. They could package these Places as they are or further customize them, adding more value. For each PlaceType there is a unit, which defines the instantiation of a unique Place.

Billing

The Place unit is the billing system. Currently the firm uses a single billing system, so until the Billing Place gets so large that a second Place makes sense, there will be only one instantiation the Billing PlaceType.

Staff Management

The Place unit is the staff member. Currently the firm has a staff of 12: six attorneys and six support staff. Each staff member has their own Place, based on the Staff Management PlaceType, which is embedded with the firm's policies and best practices regarding personnel policies, benefits, and administration. Staff member Places also serve as the personal portals for their owners. Staff members can provide ready access to the "Top Ten Frequently Asked Questions" or "Five Forms and References." They can also create rooms, controlling for private discussions.

Office/Library Management

The Place unit is the library system. There are two online research services: Lexis and Westlaw. Members of the firm might have access to one service but not another.

Client Management

The Place unit is the client. Currently the firm has 39 billing clients and 23 prospective clients. Each client Place is customized by end users to represent the unique culture and expectations of that client.

When all PlaceTypes have been set up, a Turnkey Server including all PlaceTypes is created, tested and sent to Mudge & Mudge.

We now discuss how a PlaceType is created in QuickPlace.

11.2 Creating a PlaceType

Creating a PlaceType from an existing Place is a process that involves the owner (administrator) of the Place and the QuickPlace server administrator. The required steps are:

1. The Place owner must allow the place to be published as a PlaceType and specify how it can be used (which areas to make available, whether use of PlaceBots should be allowed, and so on).
2. The QuickPlace server administrator must then create the actual PlaceType based on the prepared Place and determine where it should be listed among the other PlaceTypes available on the server.

Most often, creating a PlaceType is a four-step process where a PlaceType is created from a best practices production Place. Then a new Place is created from that PlaceType and modified to be more generic than the actual production version. Finally, a new PlaceType is created from the generic Place and made available for general use.

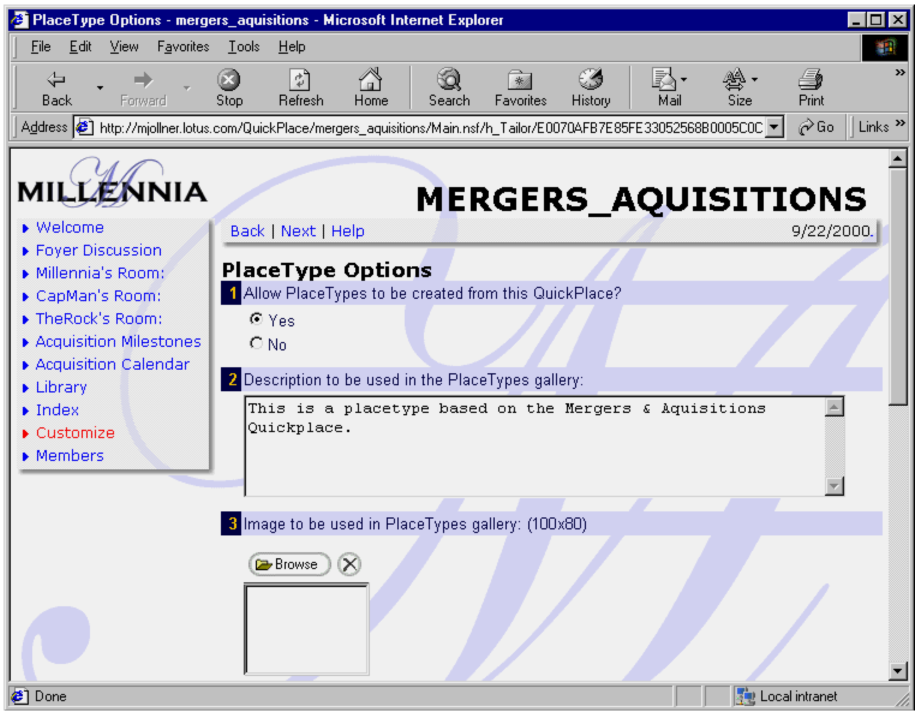
We will now walk through the actual steps in creating a PlaceType. We use 11.1.1, “Example: Millennia opens an office in Sydney, Australia” on page 321 as an illustration.

11.2.1 The Place owner part

For Millennia’s office in Sydney, we need to package all existing Places with all Placebots, but excluding all existing users. We create PlaceTypes of all existing Places, selecting to include the Placebots, but not the users.

To create a PlaceType using your Place, you have to be the manager or the owner of a Place.

1. Log in to your Place.
2. Choose **Customize** from the main menu, and click **PlaceType Options**. The PlaceType options page displays.
3. Click **edit** in the button bar. You can see the top of the PlaceType options page in edit mode in Figure 114.



http://mjollner.lotus.com/QuickPlace/mergers_aquisitions/Main.nsf/h_Tailor/E0070AFB7E85FE33052568B0005C0C

MERGERS_AQUISITIONS

9/22/2000.

PlaceType Options

1 Allow PlaceTypes to be created from this QuickPlace?

[✓ Yes] No

2 Description to be used in the PlaceTypes gallery:

```
This is a placetype based on the Mergers & Aquisitions
Quickplace.
```

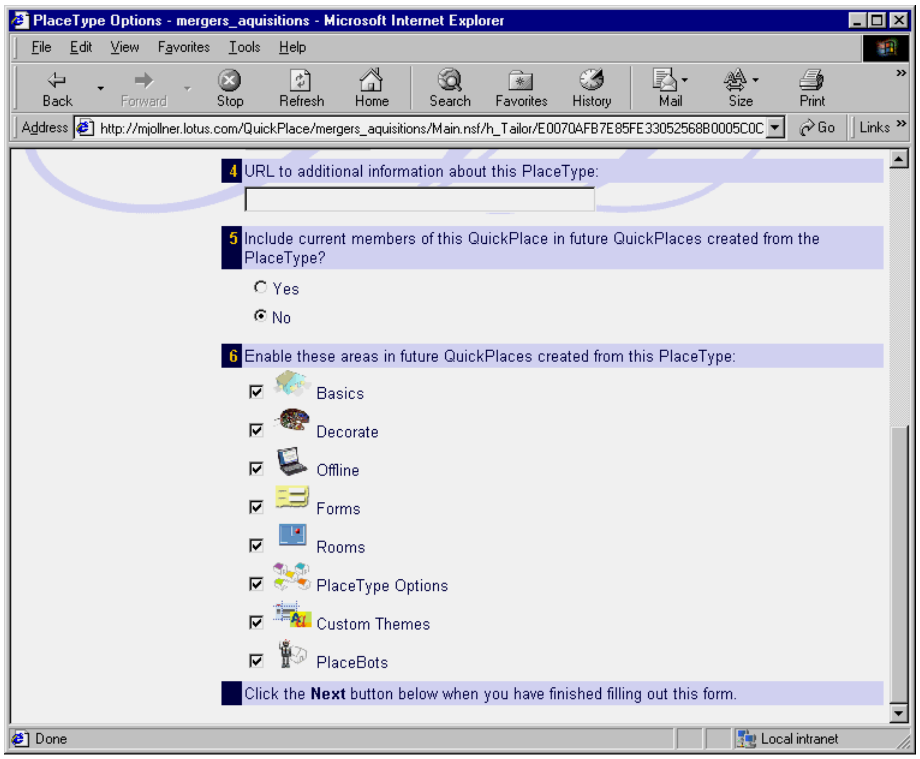
3 Image to be used in PlaceTypes gallery: (100x80)

Figure 114. The PlaceType options page

- Set **Allow PlaceTypes to be created from this QuickPlace** to **Yes**. This is the default when the page is set to edit mode the first time.
- Enter a description for your PlaceType.
- Upload a thumbnail image to represent your PlaceType using the **Image to be displayed in PlaceType Gallery** bucket (entry field).

  To create an image for the PlaceType gallery, take a screen shot of the main page of your Place and reduce it to 100 x 80 pixels.
- You can add an URL that leads to additional information about this PlaceType.
- You can choose to add all members of your Place to the PlaceType. They will automatically become members of any Place created from your PlaceType.

  We have set this option to **No** for the Millennia Places.
- Next, choose which features of your Place should be included in the PlaceType. The features included represent the customize options on the **Customize** page. By default, all components are selected as shown in Figure 115 on page 326. Deselect the ones you do not want to include by unchecking the check box next to the item.


PlaceType Options - mergers_aquisitions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print
Address http://mjollner.lotus.com/QuickPlace/mergers_aquisitions/Main.nsf/h_Tailor/E0070AFB7E85FE33052568B0005C0C
Go Links
Done
Local intranet

4 URL to additional information about this PlaceType:

5 Include current members of this QuickPlace in future QuickPlaces created from the PlaceType?

Yes [✓ No]

6 Enable these areas in future QuickPlaces created from this PlaceType:

- ✓ Basics
- ✓ Decorate
- ✓ Offline
- ✓ Forms
- ✓ Rooms
- ✓ PlaceType Options
- ✓ Custom Themes
- ✓ PlaceBots

Click the **Next** button below when you have finished filling out this form.

Figure 115. The PlaceType options page

We selected to have all features of the Millennia Places included in the PlaceType.

4. Once you are finished editing the settings for your PlaceType, click **Done**.

Your Place is now ready for turning it into a PlaceType. Notify your QuickPlace server administrator that you have created a PlaceType to be put into the PlaceType gallery.

11.2.2 The server administrator part

To copy the necessary files and make the new PlaceType available, the server administrator must log in to the administration QuickPlace and perform the following.

1. In the administration QuickPlace, click **PlaceTypes**.

   This takes you to the PlaceTypes area where the currently available PlaceTypes are listed.

2. Click **Create PlaceType...** to add a new PlaceType to the gallery.
3. A Create PlaceType form is shown.
   - Enter a name for the PlaceType.
   - Select the Place to copy from in the drop-down list.

     Only those Places whose owner has allowed to be made into PlaceTypes are shown in the list.
4. Click **Next** to create the actual PlaceType.

   QuickPlace creates a subdirectory named like the PlaceType in the AreaTypes directory and copies all databases that comprise your Place into the directory.These include:

   - Main.nsf
   - Contact1.nsf
   - All databases for Rooms created in your Place, called PageLibrary<unique identifier>.

This concludes the necessary steps to create a PlaceType.

The PlaceType is by default added as the last in the list of PlaceType. The server administrator can move it up in the list using the **Reorder** button, or make it the standard PlaceType for all Places by hiding all other PlaceTypes, using the **Show/Hide** button.

When you've finished creating a PlaceType and it has been put into the PlaceType gallery by the QuickPlace server administrator, you can use this PlaceType to create new Places, or create a Turnkey server.

Tip: If you have some PlaceTypes that you do not want your general users to be able to use, you can hide them. That way, they will not be shown to the users creating Places from the QuickPlace Welcome page, but you will still be able to use the PlaceTypes in the URL command to create a Place, as described in 8.2.17, "URL to create a Place" on page 244.

11.2.3 How to move a PlaceType to another server

Later on we discuss how to package our PlaceTypes into a Turnkey Server for new server installations, but how do you share a new PlaceTypes among *existing* QuickPlace servers?

11.2.3.1 Using QPMove

Using the QPMove utility described in Appendix C, "QuickPlace utility programs" on page 369, you can do the following

1. Create a Place based on the PlaceType you want to copy to another server.
2. Copy the directory for the newly created Place to the target server.
3. Use QPMove to finalize the copy of the Place.
4. Set up the Place on the new server as a PlaceType.

This process is a bit involved. We tried another, faster procedure that worked for us with simple PlaceTypes, but be aware that *this may not be supported and has not been subject to any further test*. We did the following

- In the AreaTypes directory in the data folder tree structure, we copied the directory for our PlaceType to the directory for Places on our target server.

  To the target QuickPlace server, the PlaceType directory now looks like a Place where the owner has allowed it to be used as a PlaceType.

- Log in as administrator in the target server and select **Create PlaceType**. The copied PlaceType will appear in the list of available PlaceTypes. Give the PlaceType a name and click **Next;** the PlaceType is now available on the target server.

Note: This procedure worked for us. However, as mentioned, it has not been tested and is not certain to work in all circumstances.

In the next section we walk through the steps to create a new Place based on a specific PlaceType.

11.3 Creating a Place based on a PlaceType

When the QuickPlace Release 2.0.x server is installed, there is only one available PlaceType called *Standard QuickPlace for Teams*. When only one PlaceType is available, all new Places will be created based on that. In this section we walk through the steps where there are several PlaceTypes to choose among when creating a new Place. We again use our Millennia example as an illustration.

To create a new Place based on a PlaceType you specify, do the following:

1. Enter the URL address of the QuickPlace server in your browser, for example:

   ```
   www.yourserver.com
   ```

   This will launch the QuickPlace Welcome page on a default-installed server.

2. Click **Create a QuickPlace**.
3. The QuickPlace server contains more than one PlaceType (see Figure 116 on page 330). Choose the **Rapid Response** PlaceType, and then click **Next**.
4. Choose a name for your Place.

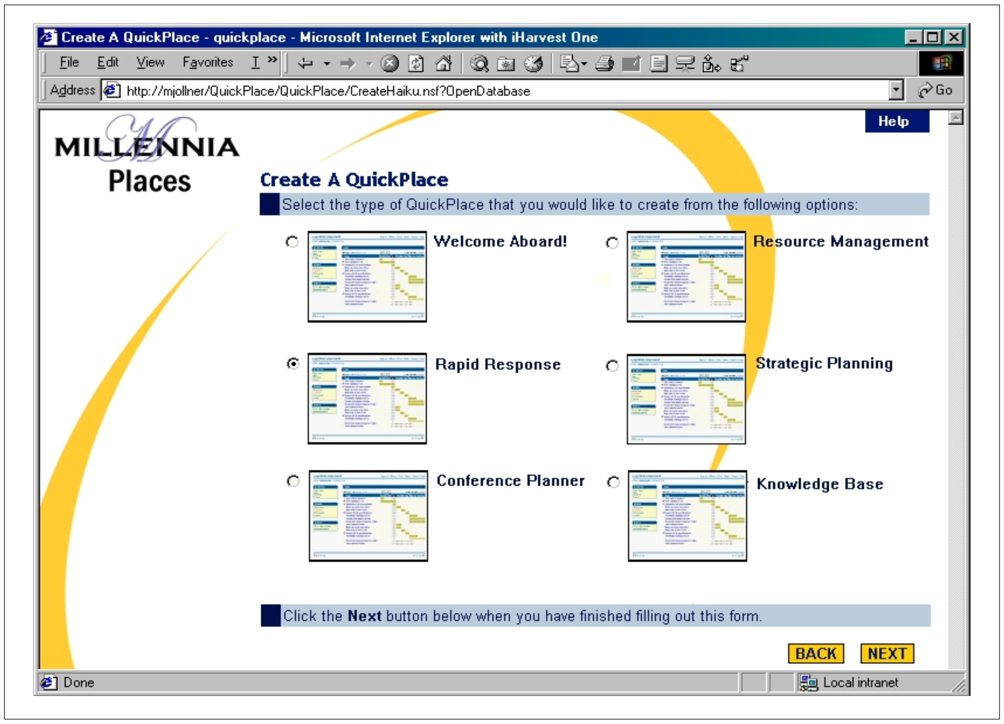
   We entered `CapMan` as the name of our Place.

   The name will be part of the Internet address for the Place, so some restrictions apply--you cannot, for example, have spaces in the Place name.
5. Enter a name, password and e-mail address for the Place manager.

   In our example, Millennia's QuickPlace server is connected to an external directory (a Domino, LDAP or NT directory), but is set up to allow people not listed in the directory (like clients and partners) to participate. However, the Place manager is in the Millennia Directory. Therefore, we enter the manager's name exactly as it appears in the directory.
6. Click **Next**; the Place is created instantaneously.

   If the PlaceType used allows anonymous users, your browser will be redirected to the Welcome page in the newly created Place. Otherwise, you will be prompted to log in to the new Place automatically.

   In either circumstance, you should log in and start adding members to your Place, or modify it further to your needs.


Create A QuickPlace - quickplace - Microsoft Internet Explorer with iHarvest One
File Edit View Favorites
Address http://mjollner/QuickPlace/QuickPlace/CreateHaiku.nsf?OpenDatabase
Go
Help
MILLENNIA Places

Create A QuickPlace

Select the type of QuickPlace that you would like to create from the following options:

( ) Welcome Aboard! ( ) Resource Management
(•) Rapid Response ( ) Strategic Planning
( ) Conference Planner ( ) Knowledge Base

Click the **Next** button below when you have finished filling out this form.

BACK
NEXT
Done
Local intranet

Figure 116. Choose a PlaceType from available options

Depending on the process/project you need to support, the new Place may be ready for use newly created “out-of-the-box”. In other cases, you need to set up things that are specific to the particular process or project you will host in your Place. In the following section, we illustrate some of the structure you may add to a new Place, by walking through an example.

11.4 Refreshing a Place with an updated PlaceType

Often PlaceTypes will be updated over time because the process they support is changed/refined, or because of bugs in a PlaceType component (forms, PlaceBots, JavaScript and so on), or to change the Theme, or for some other reason.

A PlaceType is updated by deleting the existing one and creating a new PlaceType *with the same name*.

When the updated PlaceType is available, individual Place owners can select to refresh their Place with the updates (but they do not *have* to update their Place). Refreshing is done when logged in as administrator to the Place from **Customize -> Basic -> Refresh from PlaceType**.

11.5 Adding structure to a new Place - a scenario

We will now describe a scenario with the modifications we did to the CapMan Place as a newly created Place, and as it evolved over time.

During the scenario we describe how to:

- Create a mission page
- Create Rooms for private discussion among subsets of members
- Create a link to customized help/tutorial
- Add a simple workflow for document approval

We will also refer back to some of the examples in the prior chapters as they were included in the CapMan place.

In Figure 117 you can see the structure of the Place we want to accomplish through our customization.

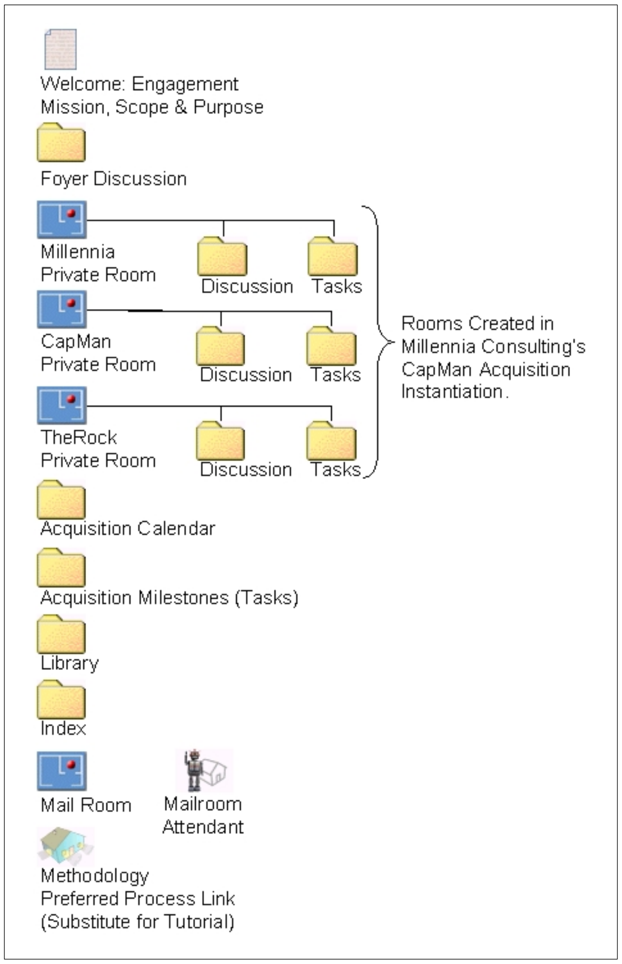


Figure 117. Structure of CapMan Acquisition Place (customized from Rapid Response PlaceType)

11.5.1 Creating a Mission Page from the Welcome Page

The purpose of the CapMan Acquisition Place is to provide a rallying point for Millennia consultants and representatives from TheRock and CapMan during the acquisition process. The Welcome Page is the first page team members see when they enter the Place and is our first chance to orient them to the way the CapMan Acquisition Place and the team functions. Our first job is to edit the Welcome Page so that the mission, communication types, and participants are all reflected on the page.

1. Enter the top-level room of the Place.
2. Click **Edit**.
3. Edit the title and/or the contents of the page. We replaced the default information with the following text:

```
Millennia Consulting's Mergers & Acquisition Team is pleased to be your
host in this shared virtual Place, created with QuickPlace, which will be
our rallying point for the acquisition of CapMan Software by TheRock, Inc.
A letter of intent to purchase was received by CapMan president Livingstone
Campbell from TheRock Legal team. Both CapMan and TheRock are very excited
about the potential of this merger and are working hard to make it a
reality.
The mission of Millennia M&A is to facilitate each phase of the acquisition
process:
     - Creative Tax and Accounting Structuring
     - Due Diligence Assistance
     - Assistance with Acquisition and Financing Documents
     - Cross-Border Interests
     - Closing Assistance
     - Post-Transaction Assistance
The Millennia M&A team is pleased to be joined by representatives from
TheRock and CapMan. For a directory of the people on this extended team,
click the Members link.
Each team has a secure room for private discussions and there is a “foyer”
for public discussions. To view content click a link on the left side of
this page. If you want to add content, click the New... button at the top of
this or any other page. QuickPlace is secure. Whenever you share
information here, you can choose which people can see or change that
information
You can check the progress of the acquisition process in Acquisition
Milestones.
For detailed information on the process used by the Millennia M&A Team, see
the M&A Process tutorial.
```

4. Click one of the following options:
   - Publish. Click Publish to publish the page immediately, without first choosing any special publication options. This option ends the editing session.
   - Publish As. Click Publish As if you want to publish the page with one or more publication options and end the editing session.
5. If you clicked Publish As in the previous step, choose one or more publication options and then click Next. QuickPlace prompts you for

additional information on each publication option. Supply the information and click Next or Done, as appropriate, until you have supplied all the necessary information.

11.5.2 Creating folders and private rooms

We need give the Place an appropriate structure to encourage the free flow of ideas within the teams from CapMan, TheRock, and Millennia. Naturally, each company may have issues that are not meant for discussion with the larger group, so we will give them each a room for private discussions as well as a "foyer" to interact with everyone else. As its name implies, the foyer is an area where issues are openly discussed among everyone who enters the Place.

To create the *Foyer Discussion*, we simply renamed the *Discussion* folder that was already created in the Place upon instantiation.

Follow these steps to rename the Folder:

1. In the sidebar, click the title of the folder, and then click the word **Folder...**
2. Enter: `Foyer Discussion` as the new title for the folder.
3. Click **Next** to begin publishing the change to the folder.
4. Click **Next**.

Using the same steps, we changed the names of *Tasks* and *Calendar* to *Acquisition Milestones* and *Acquisition Schedule*, respectively.

Now we will create a Room for Millennia consultants:

1. Make sure you are in the Home location. To get to the Home location, click **Home** in the top right corner of the QuickPlace window.
2. In the sidebar, click **Customize**.
3. Click **Rooms**.
4. Click **New Room**.
5. Enter: `Millennia's Room` as the room title.
6. Specify the title to appear in the spot in the sidebar directly under *Foyer Discussion*.
7. Click **Done**.

To create Rooms for CapMan and TheRock, repeat the previous seven steps, naming the rooms "CapMan's Room" and "TheRock's Room." Position their titles under Millennia's Room in the sidebar. Once the rooms are created, we will create a discussion folder to show the tasks folder in each room.

We wanted to give a different look and feel to these rooms, to reflect the different corporate cultures of CapMan and TheRock. TheRock's style is highly structured and conservative, while CapMan's style is innovative and sometimes unorthodox. By applying different themes to decorate these rooms, we help cue visitors to the change in their environment when they go from the "foyer" at the top level of the Place to one of the private rooms. (To decorate a room, choose **Room Options -> Decorate** and then customize the appearance of the room as you would the main part of QuickPlace—see *The QuickPlace User's Guide* for details).

After we add the members to the Place, and create groups for Millennia, CapMan, and TheRock, we will restrict access to each room so that only members of one group have access.

11.5.2.1 Members begin to add content

End users, initially the Millennia M&A Team, add content. Lots of fresh documents with descriptive titles define the utility of the place as the "go to" source for everything related to the CapMan acquisition.

11.5.2.2 QuickPlace parameters accommodate work practices

The M&A team determines that notification of the *Millennia Group* of any new editions to the *Foyer Discussion* folder is a best practice, which they will document as part of their preferred process. Unless there are unusual circumstances, authors select Notify Specific Individuals when submitting a document and then select the *Millennia Group* to notify all of the members of that group.

While this manual notification process is helpful, the M&A team is already beginning to recognize that a more fully-satisfying solution will be to customize QuickPlace to automatically handle documents, especially when they are mailed in. To handle this, they add the PlaceBot described in 7.5, "The QuickPlace Mail Room Attendant" on page 186.

11.5.3 Linking to the customized tutorial

We specify the Web address of the customized tutorial so that, when members click the word tutorial under Tools in the sidebar, the custom tutorial appears in place of the built-in tutorial. The customized tutorial could be stored in any type of Web site, but in this case is stored in another Place. The customized tutorial linked from the Rapid Response PlaceType contains a link to the standard tutorial as well as instructions on how to use features specific to this PlaceType and a preferred process for conducting business. The members of the Millennia M&A Team can change the Tutorial link to another customized tutorial that is specifically geared to customization made to the

CapMan Place and to the business processes used by the Millennia M&A Practice.

To create the link to the customized tutorial:

1. Enter the top-level room of the Place.

2. Click **Customize** in the sidebar.

3. Click **Basics**.

4. Click **Change Basics**.

5. Under “Custom Tutorial,” do one of the following:

   - If the address of the page you’re linking to is in your head or written down for you, enter the address by typing it in the box. You must begin the URL with the characters http:// or https://. For example, if the tutorial is located at www.worldclasstutorial.com, you would type:

     ```
     http://www.worldclasstutorial.com
     ```

   - If you copied the address of the page you’re linking to onto the Clipboard, paste the address into the box by clicking the box and pressing Ctrl-V.

6. Click **Done**.

11.5.4 Adding workflow for milestone approval

The Millennia M&A Team determines that a structured process is needed to facilitate the approval of acquisition milestones by CapMan and TheRock. Each milestone must be approved by the presidents of TheRock and CapMan, and once approved by both, should not be edited by either. To meet this requirement, a manager creates a new form and selects workflow options as follows:

1. Select **Customize->Forms->New Form.**
2. Select a type of form.
3. Click **Next**.
4. Select **Workflow->Modify.**
5. Select **Approval Cycle.**
6. Click **Next**.
7. Under “Who needs to approve this page?” select the following members:
   - Step 1. Select: **Helen Grayson** (President of TheRock).

- Step 2. Select: **Livingstone Campbell** (President of CapMan).
- Step 3. Select: **Helen Grayson** again.
- Step 4. Select: **Franz Kaufman** (Millennia M&A Team Leader).

8. Under “Who can edit the page after final approval?” select: **The final reviewer**.
9. Make selections for the remaining options.
10. Click **Next.**
11. Select: **Acquisition Milestones** as the destination folder for pages created with this form.

By configuring the above workflow options, we have created a means for acquisition milestones, which can be proposed by anyone to be reviewed by both companies before being placed in the *Acquisition Milestones* folder. Regardless of who proposes the milestone, both presidents must approve it before it can be published.

11.5.5 The 80 percent solution through user customization

In Figure 118 on page 338, you see the time line for the customization we are describing in our scenario.

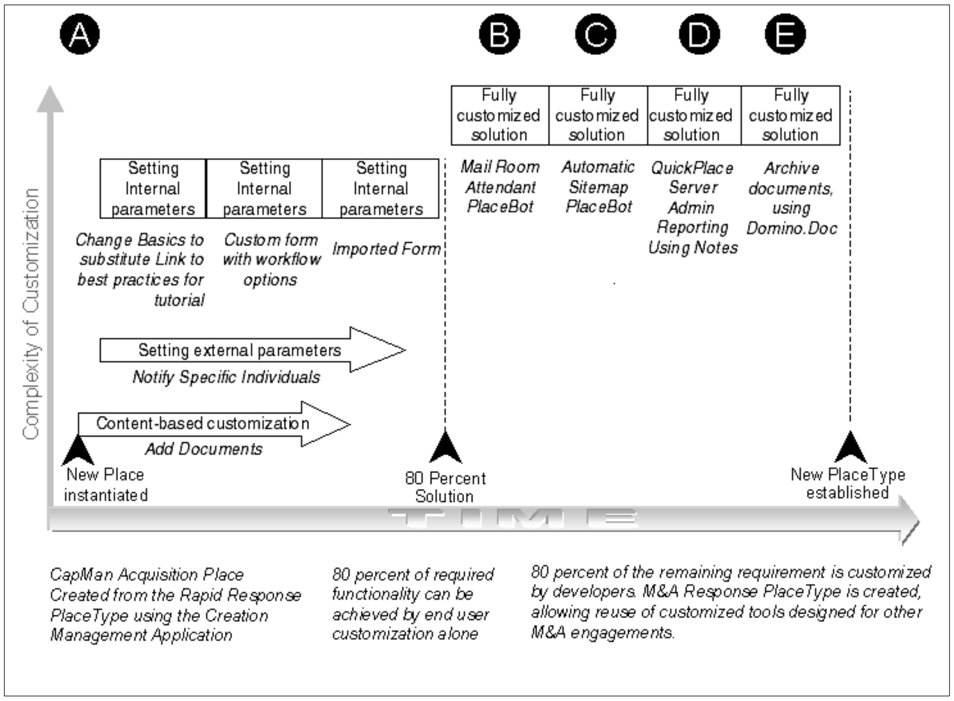

A
B
C
D
E
Fully customized solution
Fully customized solution
Fully customized solution
Fully customized solution
Setting Internal parameters
Setting Internal parameters
Setting Internal parameters
Change Basics to substitute Link to best practices for tutorial
Custom form with workflow options
Imported Form
Mail Room Attendant PlaceBot
Automatic Sitemap PlaceBot
QuickPlace Server Admin Reporting Using Notes
Archive documents, using Domino.Doc
Setting external parameters
Notify Specific Individuals
Content-based customization
Add Documents
Complexity of Customization
New Place instantiated
80 Percent Solution
New PlaceType established
TIME
CapMan Acquisition Place Created from the Rapid Response PlaceType using the Creation Management Application
80 percent of required functionality can be achieved by end user customization alone
80 percent of the remaining requirement is customized by developers. M&A Response PlaceType is created, allowing reuse of customized tools designed for other M&A engagements.

Figure 118. Millennia Consulting—TheRock's acquisition of CapMan—customization time line

The letters in the figure illustrate the following:

- A - The customization we describe in this chapter.
- B - Refer to 7.5, "The QuickPlace Mail Room Attendant" on page 186
- C - Refer to 7.4, "A site map PlaceBot" on page 171
- D - Refer to 7.6.1, "TheXRay server status reporter agent" on page 195
- E - Refer to 10.3, "Integrating with Domino.Doc" on page 287

It is widely accepted that the first part of any endeavor is relatively easy. It then gets progressively more difficult, and in the end, gets hard to do. This is known as "the 80/20 rule". The 80/20 rule is usually attributed to Vifredo Pareto (1848 - 1923), an Italian economist. The Pareto Principle states that a small number of causes is responsible for a large percentage of the effect, often a ratio of approximately 20:80.

Applying this rule, we can state that all of the customization discussed in this scenario can be accomplished by end users in relatively short order (Reference A in Figure 118).

In developing applications, we find that we can achieve 80 percent of the desired functionality in about 20 percent of the time or effort it takes to develop them, while it takes 80 percent of the time or effort to deliver the remaining 20 percent of desired functionality. In developing applications with QuickPlace, we see this principle works in our favor, as the end user can develop the 80 percent solution quickly and without any special technical knowledge.

The end user's first objective should be to achieve the 80 percent solution without intervention from professional developers. End users and Place managers (who are themselves end users) receive the base solution (in the form of a selected PlaceType) and customize it by adding content, setting external and internal parameters until they have achieved 80 percent of what they believe they need. The most we would expect a non-programmer to do would be to upload PlaceBots and skins provided in a PlaceBot library or skins/themes gallery. When they reach a point at which end-user customization can no longer meet their requirements, the end users escalate their requirement to professional developers.

The learning solution

While end users are customizing with content and by setting parameters in QuickPlace, they are also refining their requirements for the remaining 20 percent of the solution which programmers will deliver. When the requirement is handed off to the programmers, they develop a fully customized solution that delivers approximately 80 percent of the remaining 20 percent of desired functionality. If you do the math, that leaves the remaining 20 percent of the 20 percent that end users couldn't do (or 4 percent of the original requirement) left undone.

In designing and developing software solutions, what you leave undone is as important as what you do. Some activities are best completed offline from the application, either because they require face-to-face interaction or are otherwise too complex for our rudimentary ability to codify them.

To avoid creating a groupware application that inhibits people from effectively using it, developers leave room for further customizing by end users, who fine-tune the solution by content and setting parameters. By this time the requirement will have crystallized even more clearly or it will have changed direction, so it's never in anyone's interest to try to figure out the whole solution in one iteration of requirements analysis, design, development, and gap analysis.

The power of QuickPlace is that together, end user and developer collaborate in the same environment and develop solutions with less development cycle time, while allowing continuing evolution and reuse.

A new user can come into QuickPlace without even knowing what a virtual meeting place is, much less why they would want to customize one. By bundling content about why and how to do business using QuickPlace, you can teach the end user about the value of robust collaboration, document management, and workflow, while allowing unstructured and often undocumented work practices to prevail.

Customers who, before QuickPlace, would have defined their requirements by saying, "I'll know it when I see it," now can articulate requirements, because they have a baseline of experience in customizing QuickPlace. End users know when it's time to customize QuickPlace, but we developers need to know how to listen.

This concludes our Millennia Place customization scenario. We will now go further and discuss how to wrap our customizations up in a Turnkey Server.

11.6 Deploying your customized QuickPlace as a Turnkey Server

The last part of this chapter describes packaging your customized QuickPlace solutions into a self-deploying, portable, cohesive system which we call a Turnkey Server. A Turnkey Server installs ready-to-use, bundled with PlaceTypes customized to support selected communities, teams, events, and processes. It may also include additional application layers to enhance the enterprise integration and administration of QuickPlace. You can use everything you know about customizing QuickPlace to create a Turnkey Server that meets the needs of a particular customer organization or community.

In this section we discuss what makes a Turnkey Server more than just an installation disk with your company's logo on it. We will describe what we added to our test Turnkey Server and finally, we will discuss building the deployment mechanism embedded in the Turnkey Server—what you'll need to do and how you'll do it.

11.6.1 Capturing the process

A customer might point to a working ecology of applications, deployed in an organization where improved collaboration has increased innovation and improved effectiveness. The customer will tell you that's what they want. However, they often cannot distinguish between the computer applications and the human activities that accompany them. If you took the best corporate portals that have been deployed, froze them, and transplanted them to other organizations, they wouldn't take root. For the same reason that technology cannot create a team from people who have no shared interest or relationship, the portal is more than applications and content; it needs people to participate to make it live.

That's where the Turnkey Server designer come in. Like the default Welcome page in QuickPlace, which does not assume that users even know why they would want a shared virtual meeting place, each PlaceType must be embedded with the guidance for how to make the Place (and hence the key process area) work, and reflect the process or practices in its design.

For example, you could create an application that would assume the approach of a virtual interview with the new team leader and Place manager. During this interview, you could present information designed to educate the leader about how to conduct business and how to be a good facilitator, while helping the leader craft a mission and purpose for the team. Your application would then inject that information into the Welcome page of the newly instantiated place.

If you are creating a Turnkey Server as a way to deploy QuickPlace with customized PlaceTypes internally, you probably already have an existing support structure in place that also can support the processes in your PlaceTypes. On the other hand, if you are creating a Turnkey Server for use in other companies or organizations, there are much bigger requirements for having complete and self-sufficient PlaceTypes.

So, while you must consider your Turnkey Server design carefully in all cases, how much energy you should spend on it depends heavily on where the Turnkey Server will be deployed. See Appendix K.4, "Deploying QuickPlaces with a Turnkey server" on page 422 for more discussion about considerations when creating a Turnkey server.

11.7 Creating the Turnkey Server installation package

Once you have decided what is needed in a particular scenario, you can start to create a Turnkey Server that is tailored exactly to your needs. To do this, you have to go through several steps, which we explain in detail.

The items listed in Table 50 are required to create a Turnkey Server.

Table 50. Items required to create a turn-key server

Item	Description
QuickPlace CD	Make sure you have the latest release, and have it installed on the machine where your existing Places reside. You need the CD to copy the standard files for your Turnkey Server.
PlaceType(s)	Create one or more Places that reflect your customer's needs. Then create one or more PlaceTypes from those Places.
Setup.bmp (optional)	The QuickPlace CD provides the standard setup image you see when you start installing QuickPlace. You can choose to replace it by an image exactly the same size by changing that image to a *.bmp file and renaming it to setup.bmp.
Background image (optional)	You can provide your own background image. It is displayed during the install. We will explain how later in this chapter.
Additional files (optional)	Make sure you have all additional files you want to include in the Turnkey Server ready for deployment. You can select to have them copied and installed to the newly created QuickPlace automatically after installation.

11.7.1 The components in our sample Turnkey Server

The goal for our sample Turnkey Server is to wrap up some of the customizations we did in this book, and help you understand how you create your own Turnkey Server.

As you go through these descriptions, you'll find what would be discrepancies and inconsistencies if our Turnkey Server was to be held to the standards of a shipping product or intracompany solution. However, this is not the goal, so keep this in mind as you read.

We want to add the following to our Turnkey Server:

- PlaceTypes
  - Preferred Process
  - Rapid Response
  - Rapid Response for Mergers and Acquisitions
  - The 80 Percent Solution
- Places
  - Methodology - some of our PlaceTypes link to a "tutorial" that resides in the Place called Methodology, so we include this in our build as well.

- Utilities
  - qpfront.nsf - Create Place request database
- DLL hooks - event interface files
  - QPCustominvite.dll and redqptxt.nsf
  - QPLegal.dll and redqplegal.nsf
- Graphic identify
  - The background bitmap shown during installation of the Turnkey Server
- Installation utilities
  - qpmsg.exe - Program to ultimately do post-installation setup, like changing the Home URL field in the names.nsf database on the server to point the Web browser to qpfront.nsf by default, adding the QuickPlaceModules line to NOTES.INI, and so on. For our test we used a placeholder program that pops up a message box when invoked.

11.7.2 Preparations for creating a Turnkey Server

Copy all files from the QuickPlace CD into a separate directory on your machine. You will create your Turnkey Server installation CD from this directory. In our example, we called this directory QPInstall205.

Create the setup image and background image for installation. For the setup image, create an image exactly the size as the image setup.bmp found in the root directory of the QuickPlace CD. Call this image setup.bmp, to replace the existing one. On your new setup.bmp image, you *must* include the same Lotus copyright information as on the original setup.bmp. In our case we included the following information on setup.bmp:

```
Lotus QuickPlace Release 2.0.5 © 1985-2000
Lotus Development Corporation. All rights reserved.
This program is protected by U.S. and international copyright laws.
```

For the background image, create a *.bmp image with a black background. To make it universally usable, you should not make it bigger than 800 x 600 pixels, and not use more than 256 colors. We named ours MillenniaSetup.bmp. Copy both files into the QPInstall root directory.

The installation process automatically copies your PlaceTypes into the PlaceType directory on the new QuickPlace server. The same goes for other data and program files. To do this, the process must be able to find the PlaceTypes and other data files in a directory structure corresponding to the structure below the data directory of an installed QuickPlace server.

Therefore, you have to create a directory that maps to the data directory on the installed server and a directory that maps to the program directory on the installed server and place your files in those directory structures.

We created a directory named redbook-data to be our root data directory and a directory named redbook-program to hold our program files. Figure 119 shows the resulting directory structure after we added our files to the data and program directories.

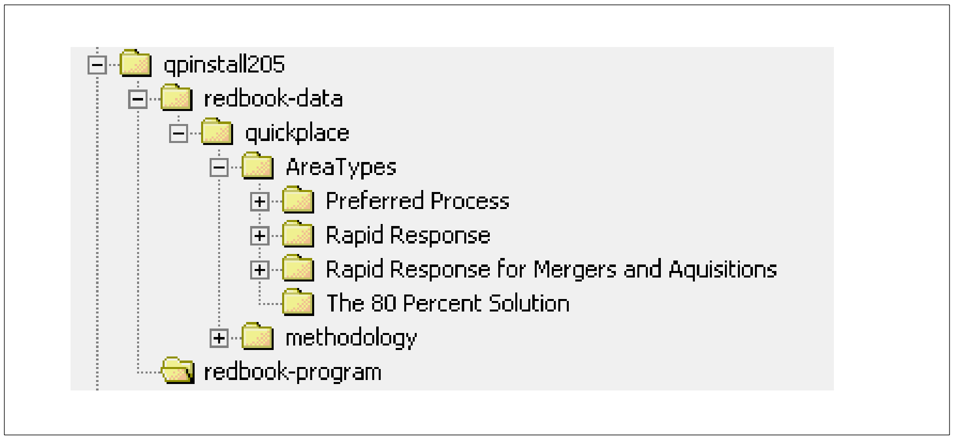


Figure 119. Installation directory structure for a QuickPlace Turnkey server

Note: The program to do the post-installation modification must be placed together with all the other QuickPlace installation files; in our case we put it in the QPInstall205 directory. It must not be placed in the directory for program files.

However, even though the post-installation program is started from the same directory as the other installation files, its working directory will be set to the QuickPlace server program directory. This is useful if your program will add some lines to QuickPlace.ini because the file can be found in the default working directory.

11.7.2.1 Change the QuickPlace.ini file

You need to change the QuickPlace.ini file that resides in the root installation directory to include your setup and background images, to copy your PlaceTypes and other files you might need for your Turnkey Server, and for any other programs that you would want to run after setup.

The following example shows the QuickPlace.ini file we used to create our Turnkey Server:

```
[InstallConfig]

# Uncomment to specify an alternative logo image. Specify the relative
# path of a BMP file.
InstallLogo=millenniasetup.bmp

# Uncomment to specify custom program files. Specify the relative path of
# a directory containing the additional files. These files will be copied
# to the user's program directory after all of the QuickPlace files. All
# of the files in the specified directory along with any subdirectories
# will be copied.
ProgramFilesDir=redbook-program

# Uncomment to specify custom data files. Specify the relative path of
# a directory containing the additional files. These files will be copied
# to the user's data directory after all of the QuickPlace files. All
# of the files in the specified directory along with any subdirectories
# will be copied.
DataFilesDir=redbook-data

# Uncomment to specify a program to be executed after the QuickPlace
# install is completed. Specify the relative path of any .EXE file.
PostInstallUtility=qpmsg.exe
```

The files we prepared for our sample Turnkey Server are available from the IBM Redbooks Web site. All you need to do is have the QuickPlace installation files and then unpack our Turnkey Server package on top of that; then you'll be ready to install our Turnkey Server. See Appendix L, "Additional Web material" on page 425 for instructions on how to get the package.

After you have made all the appropriate changes, test your Turnkey Server by installing from your installation directory. Make sure you have uninstalled any QuickPlace servers on this machine, or at least shut them down.

You may have two separate installations of QuickPlace servers on one machine, but they have to be in *completely separate directory structures and you only can run one at a time*. Both installations will use the same services on an NT machine, so do not start any QuickPlace server automatically. Instead, use the corresponding nserver.exe in the quickplace directory of the server you want to start.

When your Turnkey Server has installed, create Places based on your PlaceTypes and test them thoroughly.

Once you are satisfied with your Turnkey Server deployment, create a shipment CD by copying the complete installation directory onto it.

Now you are ready to ship your tailor-made Turnkey Server.

Summary

In the first part of this chapter, we discussed how to create and use PlaceTypes. We went through a scenario to illustrate how you can create a highly specific application from a generic PlaceType, and then moved on to a discussion of how you can package all of your customized QuickPlace functionality onto a QuickPlace Turnkey Server that is ready for redistribution within your company for sale to customers.

Appendix A. Accessing LDAP directory via SSL in QuickPlace

QuickPlace does not allow direct access over an SSL connection to a LDAP directory. The workaround for this is that you configure QuickPlace to use a Domino Directory, and configure a Directory Assistance database with an LDAP directory as a secondary directory.

You can also use this workaround to configure more than one LDAP directory for your QuickPlace server.

The configuration outlined here only allows user authentication. It does not support user look-ups in the QuickPlace security screens.

Limitation: The user names have to be unique across all directories (Domino as well as LDAP).

Tips

In a Domino environment, for all authentication purposes, Domino Directories (primary and then secondary) are searched first. Only if the user name is not found are the LDAP directories then searched.

If you have multiple Domino servers in a Domino domain, then it is suggested that all modifications to a Domino Directory be done on the replica that exists on the administration server.

Our test bed included:

- A Windows NT-based Domino server with QuickPlace as an overlay
- An AS/400-based stand-alone QuickPlace server
- A Windows NT-based SecureWay LDAP Directory server communicating only on an SSL channel
- An AS/400 LDAP server communicating on a non-SSL channel

The following configuration steps are required to configure an LDAP directory as a secondary directory for authentication. You may skip the steps your server is already configured with.

1. Create a Directory Assistance database.
2. Create an entry for LDAP directories in the Directory Assistance database.
3. Add a Directory Assistance reference to names.nsf.

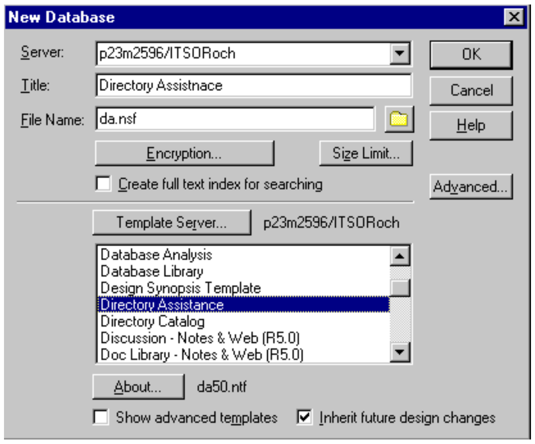
4. Establish a common certificate for SSL.
5. Configure a user directory in the QuickPlace server.
6. Initialize LDAP users in QuickPlace.

Step 1: Create a Directory Assistance database

Create a Directory Assistance database (refer to Figure 120). We used the *da50.ntf* template to create one and named it *da.nsf*. The Domino server comes with the template, but the installation process does not create the Directory Assistance database. You have to manually create and configure it.

On a standalone QuickPlace server, there is no template for a Directory Assistance database (da50.ntf), but you can copy a Directory Assistance database from another Domino server or copy the template and create a new Directory Assistance database based on the template.

We created our Directory Assistance database on the Domino server which had QuickPlace as an overlay. Once the testing proved that the LDAP Directory was accessible, we copied the same Directory Assistance database from the Domino server to the standalone QuickPlace server.



Server: p23m2596/ITSORoch
Title: Directory Assistnace
File Name: da.nsf
☐ Create full text index for searching
Template Server: p23m2596/ITSORoch
Database Analysis
Database Library
Design Synopsis Template
Directory Assistance
Directory Catalog
Discussion - Notes & Web (R5.0)
Doc Library - Notes & Web (R5.0)
da50.ntf
☐ Show advanced templates ☑ Inherit future design changes

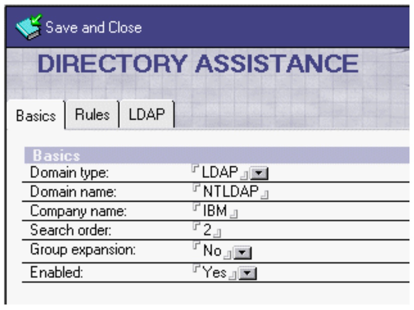
Figure 120. Creating a Directory Assistance database

Step 2: Create an entry for LDAP Directory in Directory Assistance

Open the Directory Assistance database from the Domino/Notes client. If this is the first time you are accessing the database and you see the About document screen, then close the window. You will see all the secondary directories listed (or none, if it is a new database).

Click the **Add Directory Assistance** button. This is where you supply details about the LDAP directory. Fill in the basics (refer to Figure 121). As in many Domino applications, clicking the field title displays additional help.

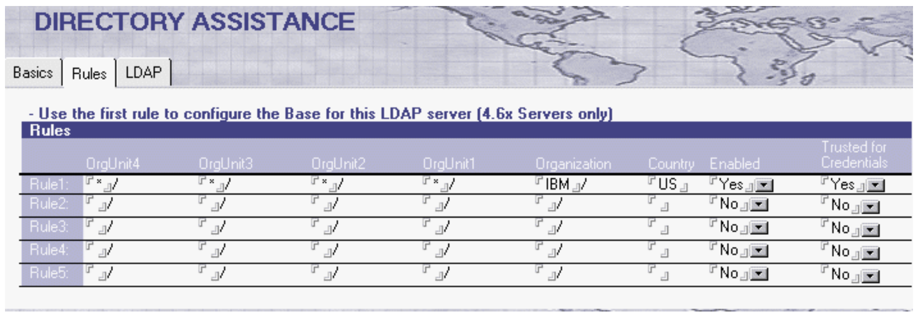
Select the Domain type to be LDAP. The domain name can be anything you wish, but it should be unique. The search order is used to tell Domino the sequence in which it should search relevant directory for the user name. This is relevant only when using multiple LDAP directories. Group expansion can be enabled only for one LDAP directory. Set the Enabled field to Yes.



Basics	
Domain type:	LDAP
Domain name:	NTLDAP
Company name:	IBM
Search order:	2
Group expansion:	No
Enabled:	Yes

Figure 121. New Directory Assistance entry for LDAP - Basics

Click the **Rules** tab. The information you enter here (refer to Figure 122) is used to limit the scope of search in the LDAP directory. We specified only one rule, which was same as our LDAP suffix. Enable the rule if you want it to be used, and set the Trusted for Credentials field to Yes.



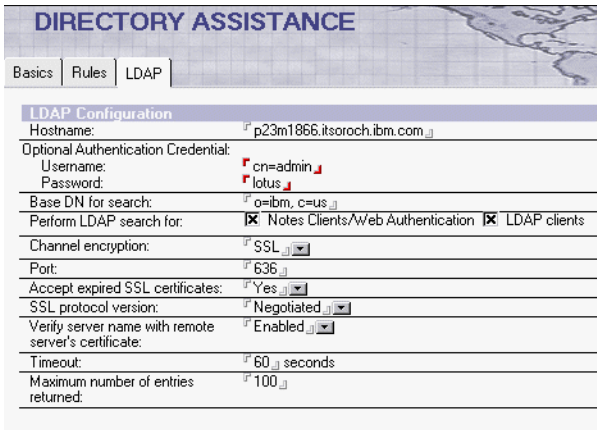
- Use the first rule to configure the Base for this LDAP server (4.6x Servers only)

Rules

	OrgUnit4	OrgUnit3	OrgUnit2	OrgUnit1	Organization	Country	Enabled	Trusted for Credentials
Rule1:	*	*	*	*	IBM	US	Yes	Yes
Rule2:							No	No
Rule3:							No	No
Rule4:							No	No
Rule5:							No	No

Figure 122. New Directory Assistance entry for LDAP - Rules

Click the **LDAP** tab. As shown in Figure 123 on page 350, this is where you specify the characteristics of the LDAP server.



DIRECTORY ASSISTANCE

Basics | Rules | LDAP

LDAP Configuration

Hostname:	p23m1866.itsoroch.ibm.com
Optional Authentication Credential:	
Username:	cn=admin
Password:	lotus
Base DN for search:	o=ibm, c=us
Perform LDAP search for:	☒ Notes Clients/Web Authentication ☒ LDAP clients
Channel encryption:	SSL
Port:	636
Accept expired SSL certificates:	Yes
SSL protocol version:	Negotiated
Verify server name with remote server's certificate:	Enabled
Timeout:	60 seconds
Maximum number of entries returned:	100

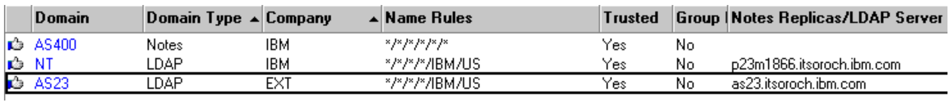
Figure 123. New Directory Assistance entry for LDAP - Configuration

You may have to contact the administrator of LDAP directory for some of this information. If your LDAP server does not allow anonymous access, then you need to provide the username and password to be used for access.

Some LDAP directories require that you specify a suffix. Use the field Base DN for Search to denote that. Both the boxes should be checked in the field Perform LDAP search for. Even though we enabled the field Verify server name with remote server's certificate, in some cases you may want to keep it disabled (less secure) if the host name does not match the server name in the certificate. This field is used only when SSL is enabled.

Then **Save and Close**.

Repeat this step if you would like to configure other LDAP Directories. Our Directory Assistance database looked as shown in Figure 124.

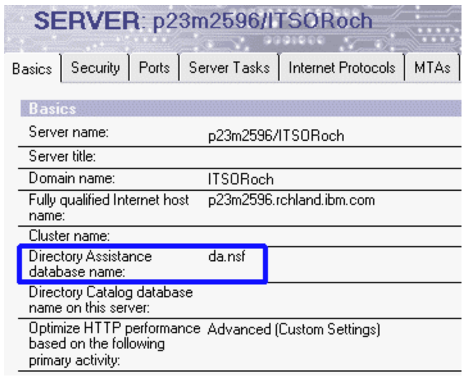


Domain	Domain Type	Company	Name Rules	Trusted	Group	Notes Replicas/LDAP Server
AS400	Notes	IBM	*/*/*/*/*	Yes	No	
NT	LDAP	IBM	*/*/*/IBM/US	Yes	No	p23m1866.itsoroch.ibm.com
AS23	LDAP	EXT	*/*/*/IBM/US	Yes	No	as23.itsoroch.ibm.com

Figure 124. View of the Directory Assistance database in our test environment

Step 3: Add Directory Assistance reference to names.nsf

Open the Domino Directory (names.nsf) on the Domino server. In the standalone QuickPlace server, open names.nsf as a local database after shutting down the server. Edit the server document to specify the Directory Assistance database (refer to Figure 125 on page 351). Save the server document.



Server name:	p23m2596/ITSORoch
Server title:	
Domain name:	ITSORoch
Fully qualified Internet host name:	p23m2596.rchland.ibm.com
Cluster name:	
Directory Assistance database name:	da.nsf
Directory Catalog database name on this server:	
Optimize HTTP performance based on the following primary activity:	Advanced (Custom Settings)

Figure 125. Directory Assistance database reference in server document

Step 4: Establish a common certificate for SSL

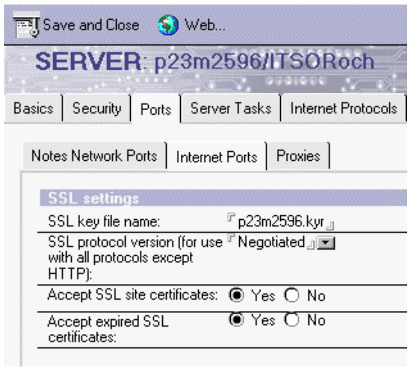
For two entities to communicate over SSL, they require a way to verify each other's authenticity. Thus, your LDAP server and the QuickPlace server (or Domino server, in the case of an overlay) should have a common certificate. For details on how to create or merge the certificates, refer to Chapter 4 in the IBM Redbook *Lotus Notes and Domino R5.0 Security Infrastructure Revealed*, SG24-5241, or to the Redpaper *Domino Certification Authority and SSL Certificates*, REDP0046 (available only in softcopy from the IBM Redbooks Web site:

```
http://ibm.com/redbooks
```

For testing purposes, we used Domino as a certificate authority to issue and stamp certificates for our SecureWay LDAP server, Domino server with QuickPlace as overlay, and standalone QuickPlace server. The LDAP Directory on AS/400 was configured without SSL and hence did not require a certificate. There are two files required for the certificate: the keyring file and the password stash file.

We placed these files in the respective data directories of the Domino server and the QuickPlace server. The server document in the names.nsf file now needed to be modified to reflect the keyring file.

Open the Domino Directory (names.nsf) on the Domino server. In the standalone QuickPlace server, open the names.nsf as a local database after shutting down the server. Edit the server document to specify the keyring file (refer to Figure 126 on page 352). Save the server document.



SERVER: p23m2596/ITSORoch

SSL settings

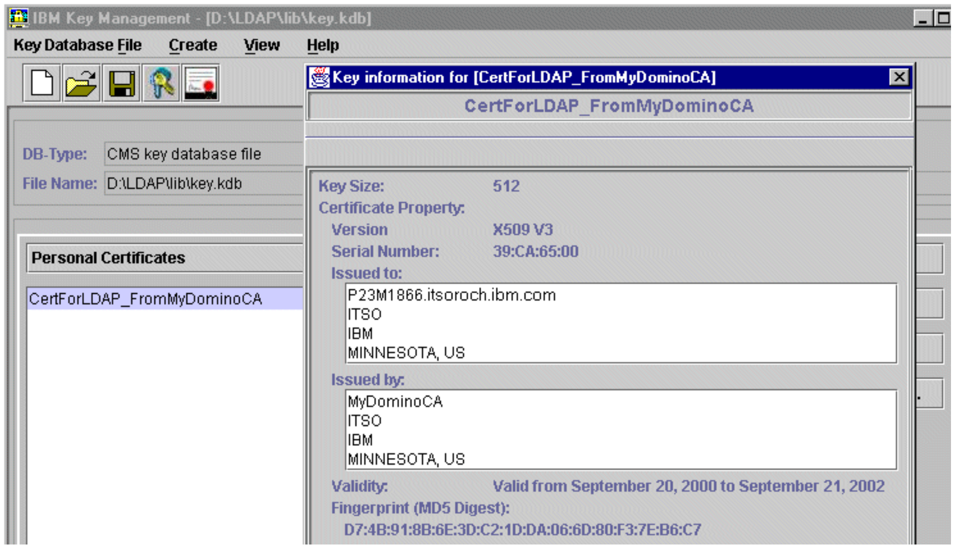
SSL key file name:	p23m2596.kyr
SSL protocol version (for use with all protocols except HTTP):	Negotiated
Accept SSL site certificates:	[✓ Yes] No
Accept expired SSL certificates:	[✓ Yes] No

Figure 126. Specify the keyring file in server document

The SecureWay LDAP could be configured in three modes: non-SSL, SSL only, and both. For our testing purposes, we configured it in *SSL only* mode to assure that LDAP communication takes place only over an SSL connection.

If you configure the LDAP server in dual mode, the Directory Assistance database entry for the LDAP server will dictate in which mode Domino will communicate with the LDAP server.

On the SecureWay LDAP Directory server in our test environment, the SSL certificate information is displayed as shown in Figure 127 on page 353 and the LDAP configuration looked as shown in Figure 128 on page 353.

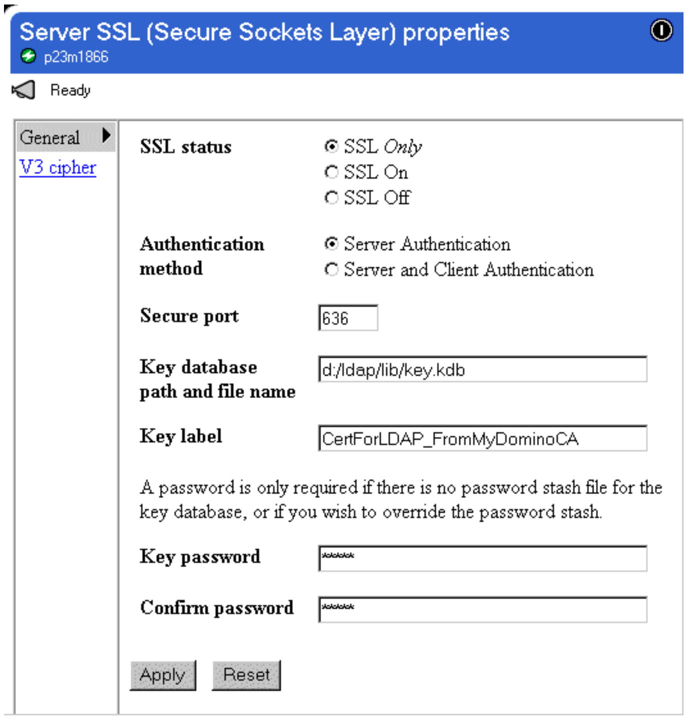

IBM Key Management - [D:\LDAP\lib\key.kdb]
Key Database File
Create
View
Help
Key information for [CertForLDAP_FromMyDominoCA]
CertForLDAP_FromMyDominoCA

DB-Type: CMS key database file
File Name: D:\LDAP\lib\key.kdb

Personal Certificates
CertForLDAP_FromMyDominoCA

Key Size: 512
Certificate Property:
Version X509 V3
Serial Number: 39:CA:65:00
Issued to:
P23M1866.itsoroch.ibm.com
ITSO
IBM
MINNESOTA, US
Issued by:
MyDominoCA
ITSO
IBM
MINNESOTA, US
Validity: Valid from September 20, 2000 to September 21, 2002
Fingerprint (MD5 Digest):
D7:4B:91:8B:6E:3D:C2:1D:DA:06:6D:80:F3:7E:B6:C7

Figure 127. SSL certificate information of our test SecureWay LDAP Directory server



SSL status: [✓ SSL Only] SSL On SSL Off

Authentication method: [✓ Server Authentication] Server and Client Authentication

Secure port: 636

Key database path and file name: d:/ldap/lib/key.kdb

Key label: CertForLDAP_FromMyDominoCA

A password is only required if there is no password stash file for the key database, or if you wish to override the password stash.

Key password: *****

Confirm password: *****

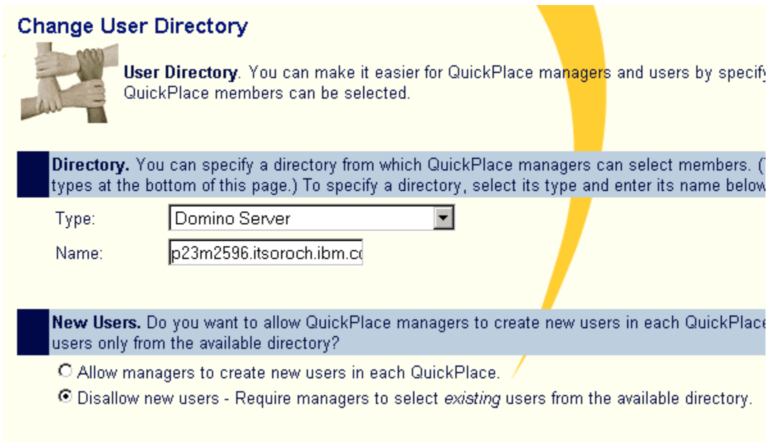
Figure 128. SSL setting in SecureWay Directory

Step 5: Configure user directory in QuickPlace server

We now have to tell QuickPlace which directory to use for user lookups and authentication. Go to the QuickPlace server's main screen (Welcome page). Click **Sign In** at the top right-hand corner. Enter the QuickPlace server administrator's user ID and password.

Click **Server Settings**, then click **User Directory**. Click the **Change Directory** button at the bottom right-hand part of the screen. You will see a screen similar to Figure 129. Select **Domino Directory as Type** and in the Name field, enter the host name for the Domino server. In the case of a standalone QuickPlace, enter the name of the QuickPlace server.

We decided not to allow any new users, in order to ensure that authentication takes place only with the users present in the LDAP server directory.



Change User Directory

User Directory. You can make it easier for QuickPlace managers and users by specify
QuickPlace members can be selected.

Directory. You can specify a directory from which QuickPlace managers can select members. (
types at the bottom of this page.) To specify a directory, select its type and enter its name below

Type:	Domino Server
Name:	p23m2596.itsoroch.ibm.c

New Users. Do you want to allow QuickPlace managers to create new users in each QuickPlace
users only from the available directory?

○ Allow managers to create new users in each QuickPlace.

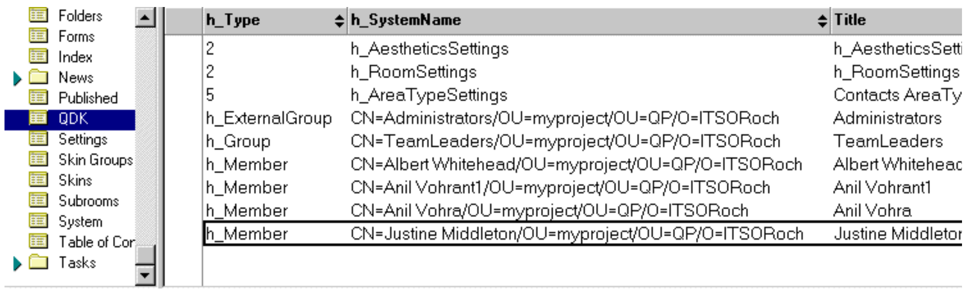
● Disallow new users - Require managers to select *existing* users from the available directory.

Figure 129. Setting user directory in QuickPlace

Step 6: Initialize users in QuickPlace

Shut down the Domino server or QuickPlace server on which your QuickPlace resides, or do a filecopy at the operating system level of Contacts1.nsf of that QuickPlace.

Open the Contacts1.nsf of that QuickPlace as a local database in the Domino/Notes client. Go to the *QDK* view (refer to Figure 130 on page 355). You see all the members listed there. Each entry is a regular Notes document. Select any h_Member type of document, and copy and paste right back into the view.



h_Type	h_SystemName	Title
2	h_AestheticsSettings	h_AestheticsSetti
2	h_RoomSettings	h_RoomSettings
5	h_AreaTypeSettings	Contacts AreaTy
h_ExternalGroup	CN=Administrators/OU=myproject/OU=QP/O=ITSORoch	Administrators
h_Group	CN=TeamLeaders/OU=myproject/OU=QP/O=ITSORoch	TeamLeaders
h_Member	CN=Albert Whitehead/OU=myproject/OU=QP/O=ITSORoch	Albert Whiteheac
h_Member	CN=Anil Vohrant1/OU=myproject/OU=QP/O=ITSORoch	Anil Vohrant1
h_Member	CN=Anil Vohra/OU=myproject/OU=QP/O=ITSORoch	Anil Vohra
h_Member	CN=Justine Middleton/OU=myproject/OU=QP/O=ITSORoch	Justine Middletor

Figure 130. View of all users

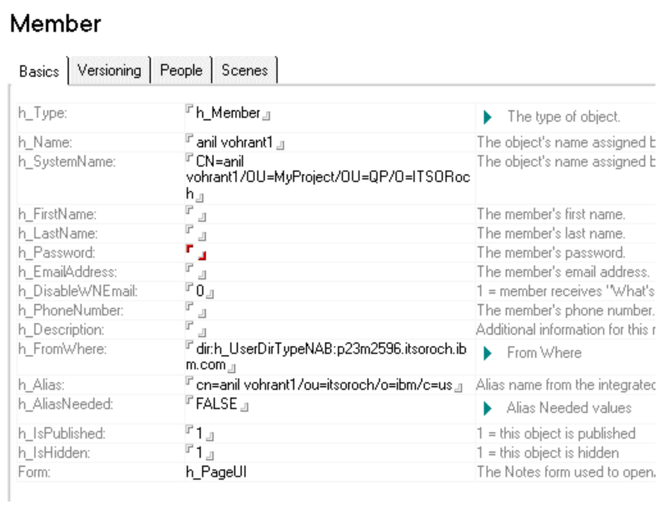
Open the newly pasted h_Member type document (refer to Figure 131 on page 356). Go into edit mode (CTRL-E). The fields you need to modify are h_Name, h_SystemName, h_Password, h_EmailAddress and h_Alias. Simply replace the common name part of the user in h_SystemName with the new user you have in mind.

Blank the password out. Enter the valid e-mail for the new user. The h_Alias should be the fully qualified name of the user as it exists in the LDAP directory. Use a forward slash (/) and not a comma (,) to separate the name components. Save the document.

You have to know beforehand the e-mail and user IDs of the people you would like to use in the QuickPlace from LDAP directory. The user ID (h_name field) should be unique in the environment. QuickPlace scans for the names in the Domino environment first, and then in the LDAP environment.

Remember to add the h_SystemName of this new user to the h_members group of the Place in the Domino Directory (names.nsf). For our test example, the QuickPlace was MyProject and that group in the names.nsf was h_members/MyProject/QP/ITSORoch/IBM.

The last part in the h_FromWhere field should reflect the Domino or QuickPlace server address that you configured as a user directory in QuickPlace.



h_Type:	h_Member	The type of object.
h_Name:	anil vohrant1	The object's name assigned b
h_SystemName:	CN=anil vohrant1/OU=MyProject/OU=QP/O=ITSORoch	The object's name assigned b
h_FirstName:		The member's first name.
h_LastName:		The member's last name.
h_Password:		The member's password.
h_EmailAddress:		The member's email address.
h_DisableWNEmail:	0	1 = member receives "What's
h_PhoneNumber:		The member's phone number.
h_Description:		Additional information for this r
h_FromWhere:	dir:h_UserDirTypeNAB:p23m2596.itsoroch.ibm.com	From Where
h_Alias:	cn=anil vohrant1/ou=itsoroch/o=ibm/c=us	Alias name from the integratec
h_AliasNeeded:	FALSE	Alias Needed values
h_IsPublished:	1	1 = this object is published
h_IsHidden:	1	1 = this object is hidden
Form:	h_PageUI	The Notes form used to open.

Figure 131. h_Member type document in Contact1.nsf

Tip

When creating a new Place (make sure the Domino or QuickPlace server is running), you can use the user name and password of any user in the LDAP directory in the Create New QuickPlace form. If the user name and password is valid, Domino (or the QuickPlace server) will authenticate with the LDAP directory and create the new Place with the given user marked as QuickPlace administrator - just like creating any other Place.

If you used filecopy for Contacts1.nsf, rename the copy on the server for backup. Copy the modified version of the file back to the server, overwriting the existing Contacts1.nsf.

Once the names are added, they will show up for selection in all the security screens of that QuickPlace.

Note: We deleted the h_password field from all the member records, referring to the LDAP users, in contacts1.nsf. We did this to ensure that Domino will use the user credentials in the LDAP server's database, and not those from its own cache.

Once you set this up, all user authentication happens without any issues.

If you are planning to use the same group of people again and again, then you may want to save the QuickPlace as a PlaceType, along with all the member information.

Appendix B. QuickPlace for AS/400 considerations

This appendix gives a brief summary of special considerations regarding installing and configuring Lotus QuickPlace server on an AS/400. For detailed information, see *Installing and Managing QuickPlace for AS/400*. It is available on the Web at:

`http://www.as400.ibm.com/quickplace`

Also refer to the Redpaper *Lotus QuickPlace for AS/400: Setup and Management Considerations.* It is available at:

`http://www.redbooks.ibm.com`

B.1 Installation and configuration

Installation of the QuickPlace code is done using the Load and Run (LODRUN) command.

Setting up a new QuickPlace server and its configuration, in stand-alone or in overlay mode, can be done by using Operations Navigator or with AS/400 commands. A QuickPlace plug-in is available to support the Operations Navigator environment. You may set up only one stand-alone QuickPlace server per AS/400. As an overlay, you can set up one QuickPlace server per Domino server on any number of Domino servers running on an AS/400.

Figure 132 on page 360 shows how to launch a *New QuickPlace Server* setup from Operations Navigator. The setup launches the browser, and all the subsequent steps during the setup are browser-based.

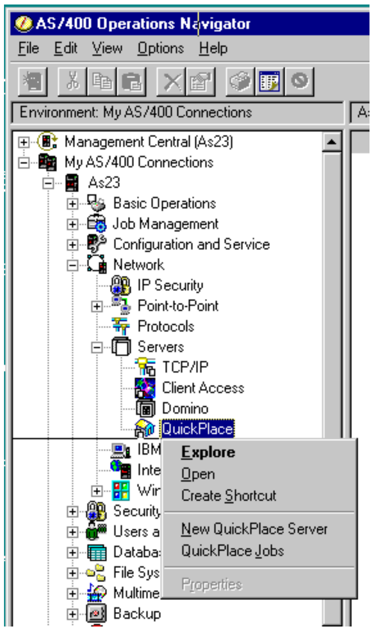
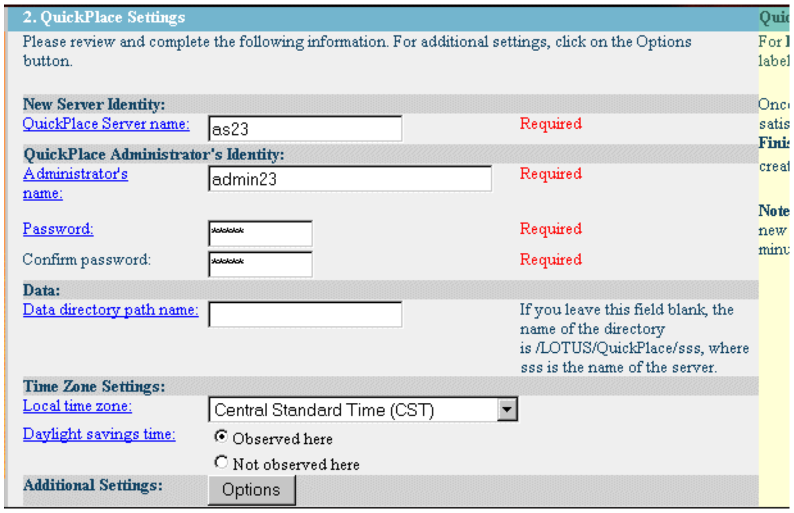


Figure 132. Setup new QuickPlace server via Operations Navigator

The first screen presented during the setup allows you to create a new QuickPlace server (stand-alone), or add QuickPlace to an existing Domino server (overlay). Figure 133 and Figure 134 on page 361 show you, as a sample, the configuration parameters that you can specify in stand-alone mode. When running other jobs that use TCP/IP on AS/400, it is recommended that you specify the IP address (bind QuickPlace server to an IP address) in the *Additional Settings* screen when installing QuickPlace server in stand-alone mode.

In overlay mode, you bind the HTTP port by setting the *Bind to host name* field in the server document under the **Internet Protocols->HTTP** tab. For details about certifier IDs, refer to 3.1.3, “Certifier ID considerations and certifier hierarchy” on page 25.



2. QuickPlace Settings

Please review and complete the following information. For additional settings, click on the Options button.

New Server Identity:

QuickPlace Server name:	as23	Required

QuickPlace Administrator's Identity:

Administrator's name:	admin23	Required
Password:	*****	Required
Confirm password:	*****	Required

Data:

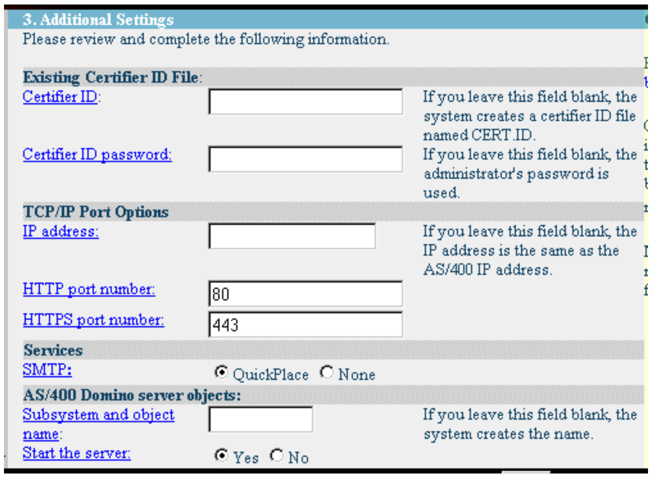
Data directory path name:		If you leave this field blank, the name of the directory is /LOTUS/QuickPlace/sss, where sss is the name of the server.

Time Zone Settings:

Local time zone:	Central Standard Time (CST)
Daylight savings time:	[✓ Observed here] Not observed here

Additional Settings: Options

Figure 133. Configuration parameter screen for QuickPlace server setup



3. Additional Settings

Please review and complete the following information.

Existing Certifier ID File:

Certifier ID:		If you leave this field blank, the system creates a certifier ID file named CERT.ID.
Certifier ID password:		If you leave this field blank, the administrator's password is used.

TCP/IP Port Options

IP address:		If you leave this field blank, the IP address is the same as the AS/400 IP address.
HTTP port number:	80	
HTTPS port number:	443	

Services

SMTP:	[✓ QuickPlace] None

AS/400 Domino server objects:

Subsystem and object name:		If you leave this field blank, the system creates the name.
Start the server:	[✓ Yes] No	

Figure 134. Additional settings during QuickPlace server setup

When using AS/400 commands, the *Add QuickPlace to Domino* (ADDLQPDOM) command (see Figure 135 on page 362) is used to set up the QuickPlace server as an overlay. In our case, *qpres24* was our Domino server.

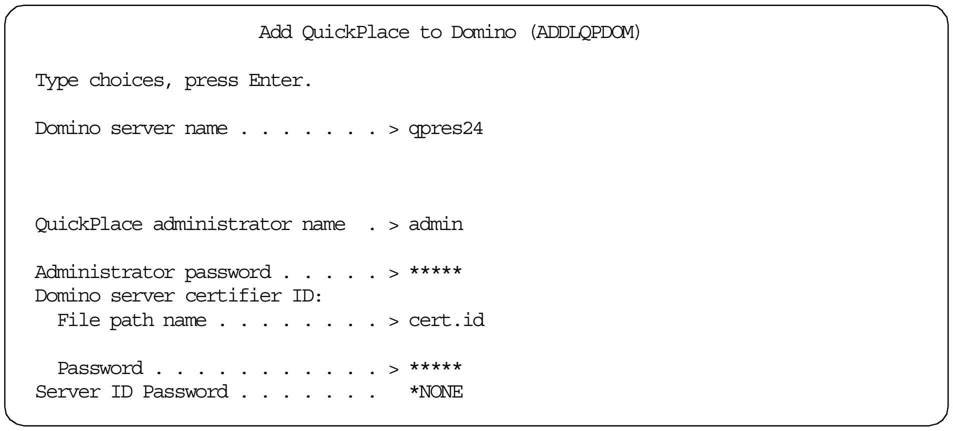
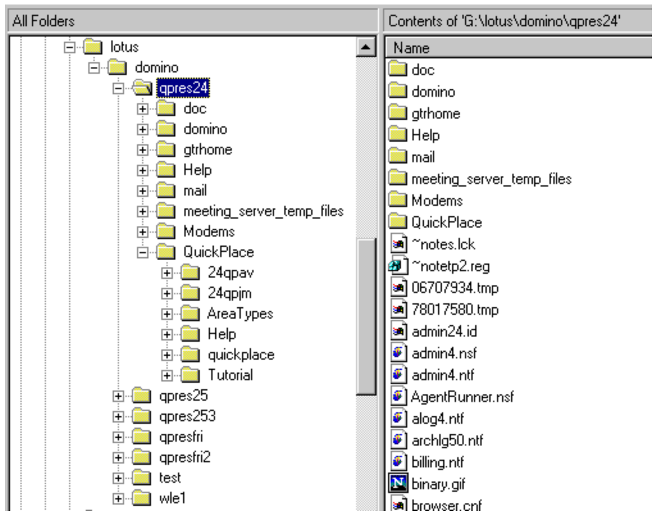

```
                         Add QuickPlace to Domino (ADDLQPDOM)

 Type choices, press Enter.

 Domino server name . . . . . . . > qpres24



 QuickPlace administrator name  . > admin

 Administrator password . . . . . > *****
 Domino server certifier ID:
   File path name . . . . . . . . > cert.id

   Password . . . . . . . . . . . > *****
 Server ID Password . . . . . . .   *NONE
```

Figure 135. ADDLQPDOM command for overlay QuickPlace server

For additional QuickPlace-related AS/400 commands, refer to *Installing and Managing QuickPlace for AS/400.*

B.2 Directory structure

When a QuickPlace server is installed as an overlay (on top of a Domino server), the data files associated with the QuickPlace server are under Domino's data directory. In our case, that was *Lotus/Domino/qpres24*. *qpres24* was our Domino server name. All the user-created QuickPlaces were under the *QuickPlace* directory. As you can see in Figure 136 on page 363, we had multiple Domino (partitioned) servers configured. We had created two test QuickPlaces: 24qpav and 24qpjm.

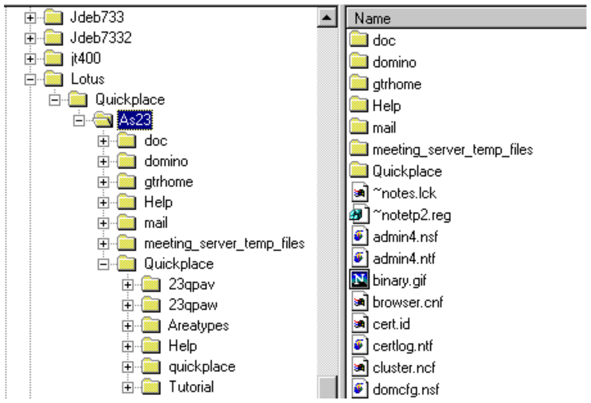


```
lotus
  domino
    qpres24
      doc
      domino
      gtrhome
      Help
      mail
      meeting_server_temp_files
      Modems
      QuickPlace
        24qpav
        24qpjm
        AreaTypes
        Help
        quickplace
        Tutorial
    qpres25
    qpres253
    qpresfri
    qpresfri2
    test
    wle1
```

```
doc
domino
gtrhome
Help
mail
meeting_server_temp_files
Modems
QuickPlace
~notes.lck
~notetp2.reg
06707934.tmp
78017580.tmp
admin24.id
admin4.nsf
admin4.ntf
AgentRunner.nsf
alog4.ntf
archlg50.ntf
billing.ntf
binary.gif
browser.cnf
```

Figure 136. QuickPlace server directory structure on AS/400 in overlay mode

When installed in stand-alone mode, the QuickPlace server files are in the */lotus/quickplace/<ServerName>* directory by default. In our case, the QuickPlace server name was *as23*. The directory structure is shown in Figure 137. We had created two test QuickPlaces: 23qpav and 23qpaw.



```
Jdeb733
Jdeb7332
jt400
Lotus
  Quickplace
    As23
      doc
      domino
      gtrhome
      Help
      mail
      meeting_server_temp_files
      Quickplace
        23qpav
        23qpaw
        Areatypes
        Help
        quickplace
        Tutorial
```

```
doc
domino
gtrhome
Help
mail
meeting_server_temp_files
Quickplace
~notes.lck
~notetp2.reg
admin4.nsf
admin4.ntf
binary.gif
browser.cnf
cert.id
certlog.ntf
cluster.ncf
domcfg.nsf
```

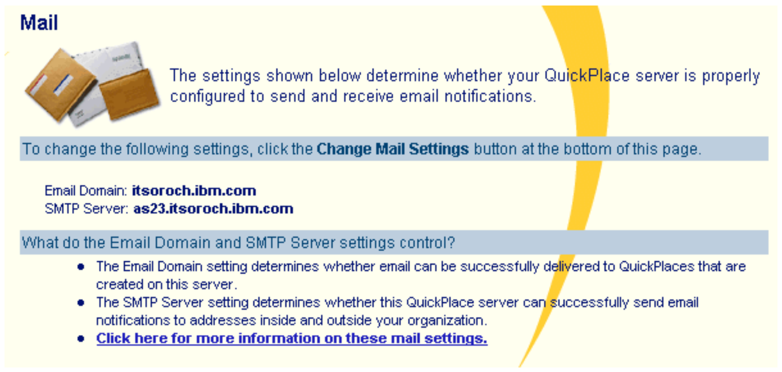
Figure 137. QuickPlace server directory structure on AS/400 in stand-alone mode

The program files for QuickPlace server are in the *QSys.Lib/Qqplace.Lib* directory. The *QIBM/ProdData/Lotus/QuickPlace* directory contains java classes, fonts, setup components, short-cuts to programs, etc. The Web-based QuickPlace setup is under the *QIBM/UserData/HTTPSetup* directory.

For demonstration purposes, we had mapped the AS/400 directories as network drives in our Windows NT workstation client.

B.3 Mail

A stand-alone QuickPlace server can use either its own (built-in) SMTP service, or any other SMTP server on the network, including Domino. If AS/400 has the SMTP server, then the QuickPlace server can leverage that service. For example, as shown in Figure 138, we configured our stand-alone QuickPlace server (as23) to use the SMTP service of the host AS/400 server.


Mail

The settings shown below determine whether your QuickPlace server is properly configured to send and receive email notifications.

To change the following settings, click the **Change Mail Settings** button at the bottom of this page.

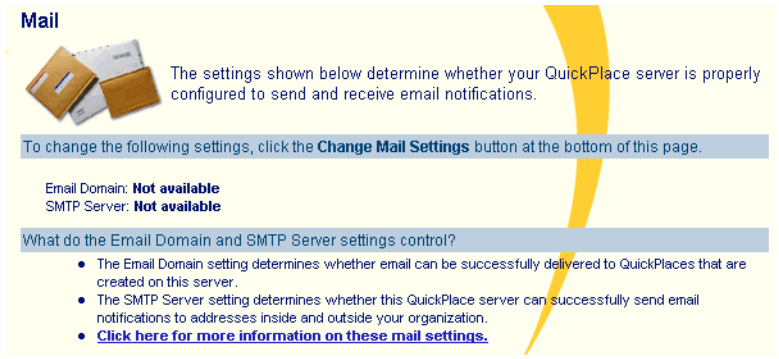
Email Domain: **itsoroch.ibm.com**
SMTP Server: **as23.itsoroch.ibm.com**

What do the Email Domain and SMTP Server settings control?

- The Email Domain setting determines whether email can be successfully delivered to QuickPlaces that are created on this server.
- The SMTP Server setting determines whether this QuickPlace server can successfully send email notifications to addresses inside and outside your organization.
- **Click here for more information on these mail settings.**

Figure 138. Sample of stand-alone QuickPlace server configured to external SMTP server

In overlay mode, the QuickPlace server must use the SMTP service as configured for its host Domino server. If the Domino server is using SMTP, so must QuickPlace. In overlay mode, the mail settings in the QuickPlace server cannot be changed (see Figure 139 on page 365).



The settings shown below determine whether your QuickPlace server is properly configured to send and receive email notifications.

To change the following settings, click the **Change Mail Settings** button at the bottom of this page.

Email Domain: **Not available**
SMTP Server: **Not available**

What do the Email Domain and SMTP Server settings control?

- The Email Domain setting determines whether email can be successfully delivered to QuickPlaces that are created on this server.
- The SMTP Server setting determines whether this QuickPlace server can successfully send email notifications to addresses inside and outside your organization.
- **Click here for more information on these mail settings.**

Figure 139. Mail configuration for overlay QuickPlace server

Note

When Domino for AS/400 was released, a single AS/400 system could only support a Domino server using either OS/400 SMTP or Domino SMTP, but not both. If there was a need for SMTP on the AS/400 system because of something other than Domino (such as OfficeVision/400), Domino servers would also have to use the OS/400 SMTP.

PTFs are now available to allow the coexistence of the two SMTP stacks. The function is bind to specific IP address and it means that both SMTP stacks (AS/400 and Domino) are supported at the same time. This is achieved by new functionality that allows the OS/400 SMTP Server and OS/400 SMTP Client to each bind to a specific TCP/IP address based on data from specific data areas.

With bind to specific IP address support, Domino for AS/400, with Domino SMTP support, can coexist with OS/400 SMTP. OS/400 SMTP can be used by both OfficeVision/400 users and Domino for AS/400 users with AnyMail Integration support (configured on a single Domino server on any given AS/400 system). This means that you can have multiple Domino servers with Domino SMTP, as well as one Domino server configured with AnyMail Integration support (OS/400 SMTP) on the same AS/400 system at the same time.

Bind to specific IP address functionality is implemented on AS/400 systems at V4R2M0 or later with PTF support. See *Lotus Domino for AS/400: Internet Mail Topics,* SG24-5990 for more information on the PTFs and how to configure this support.

B.4 Other considerations

Here are some things to keep in mind when using QuickPlace for AS/400. For more information, refer to *Installing and Managing QuickPlace for AS/400*.

Utilities

When using QPMove and QPUser utilities on AS/400, there are a number of steps to be followed to execute them successfully. For details see Chapter 12 in *Installing and Managing QuickPlace for AS/400*.

Customizing and programming

When customizing the QuickPlace environment via PlaceBots or APIs that access the system's file structure, care should be taken to specify forward slash (/) instead of backward slash (\) as the directory delimiter, in order for the customization to be portable on AS/400 and Windows NT. It is recommended that a run-time check in the program for the operating environment should determine the appropriate delimiter to be used.

We tested the Mapperizer example on AS/400. If you want to enable debug mode in the Mapperizer PlaceBot, then you should change the path of the output file appropriately. In normal mode, the debug file is not used.

We tested the MailRoomAttendant sample PlaceBot in the AS/400 environment without any issues. For this PlaceBot to work in your environment, you may have to change the LotusScript file for directory path. In the QuickPlace for AS/400 environment, beginning the directory path with "/" overrides the data directory being default. So if you want to refer to files and directories relative to (within) the data directory of QuickPlace, do not begin the path with "/" ; otherwise, you must supply the full path.

Note: For some QDK samples to work in an AS/400 environment, you may have to modify them to use a forward slash (/) as the file delimiter.

File attachments

The size of the attachments to a page are limited to 16 MB.

Fonts

Fonts may appear different when running a QuickPlace server on AS/400, as compared to a QuickPlace server on Windows NT.

Chat

Chat (Sametime server) can be enabled on only one QuickPlace server (stand-alone or overlay) on an AS/400 system.

External directories

QuickPlace for AS/400 does not support Windows NT Domain as an external directory. LDAP Directory or Domino Directory are valid configurations. The *No Directory* option lets you configure the QuickPlace server to use and manage its own internal Directory.

Appendix C. QuickPlace utility programs

This appendix gives a brief description of the two QuickPlace utility programs:

- QPMove
- QPUser

They are available for download from the *Goodies : Utilities* section of the QuickPlace DevZone at:

`http://www.quickplace.com/devzone`

C.1 QPMove Release 2.0

QPMove, the QuickPlace Move utility, doesn't actually move QuickPlaces. QPMove modifies QuickPlaces after you have:

- Moved a QuickPlace to another server
- Copied a QuickPlace to another server
- Renamed a server
- Renamed a QuickPlace

QPMove works with all releases of QuickPlace up to and including release 2.0.5. For example, you can use QPMove Release 2.0 to move a QuickPlace you created with QuickPlace Release 1.0x. Note, however, that the QuickPlace version used to create the QuickPlace you want to move or copy must match the version of QuickPlace running on the destination server.

Note: You cannot use QPMove to move a QuickPlace on a partitioned Domino server to another location on the server or to another server.

C.1.1 What happens to the QuickPlace URL

When you move, copy, or rename a QuickPlace, the URL (or address) for the QuickPlace changes. For example, suppose you have a QuickPlace called Acme on a server named Sales, and the URL for the QuickPlace is http:\sales.yourdomain\Acme. If you move QuickPlace Acme to a server called Demo, the URL changes to http:\demo.yourdomain\Acme.

Now suppose that you want to rename QuickPlace Acme on server Sales, but you don't want to move it. If you use QPMove to change the QuickPlace name Acme to Acme2000, the URL for the QuickPlace changes from http:\sales.yourdomain\Acme to http:\sales.yourdomain\Acme2000.

For more information on how to move, copy, or rename a QuickPlace, as well as how to use the QPMove utility, refer to the QPMove utility documentation.

C.2 QPUser Release 2.0

QPUser, the QuickPlace Change-Password utility, lets a QuickPlace server administrator assign new passwords to QuickPlace members who have forgotten their passwords and are therefore unable to sign in to their QuickPlace.

Note: You cannot use QPUser to assign a new password to a QuickPlace member if the QuickPlace to which that member belongs resides on a partitioned Domino server.

The QPUser utility has different parameters that allow you to list all members in a QuickPlace, change passwords with or without confirmation prompts, and so on.

Appendix D. Domino URL commands

This information has been taken from the online help in Domino R5.0.

D.1 Domino URL command syntax structure

A URL command combines a specific URL with a command that manipulates the item. Adding Domino URL commands as HTML in forms gives users shortcuts for navigating databases and performing tasks quickly.

Domino URL commands have the syntax:

```
http://Host/DominoObject?Action&Arguments
```

Where

Host = a DNS entry or an IP address, like *www.lotus.com*

DominoObject = a Domino construct (For example, a database, view, document, form, navigator, agent, and so on.)

Action = the desired operation on the specified Domino object (For example, ?OpenDatabase, ?OpenView, ?OpenDocument, ?EditDocument, ?OpenForm, and so on.)

Arguments = a qualifier of the action. (For example, Count = 10 combined with the ?OpenView action limits the number of rows displayed in a view to 10.)

D.1.1 Syntax guidelines

Special identifiers used in Domino URL commands include: $defaultView, $defaultForm, $defaultNav, $searchForm, $file, $icon, $help, $about, and $first.

DominoObject can be any of the following: for a database, the database name or replica ID; for other objects, the Domino object's name, universal ID, NoteID or special identifier. For example, to specify a view in a URL, you can use any of the following: the view name, view universal ID, view NoteID, or $defaultView.

A Domino object's name and universal ID are identical in all replicas of a database, but the Domino object's NoteID will probably change in database replicas. Therefore, it is best to use the Domino object name or universal ID in URLs. One name or alias can refer to two objects -- for example, two forms with the same name when one is hidden from Notes users and one is hidden from Web users.

Action can be explicit or implicit. Examples of explicit actions include ?OpenServer, ?OpenDatabase, ?OpenView, ?OpenDocument, ?OpenForm, and ?EditDocument. Examples of implicit actions include ?Open, ?Edit, and ?Delete. If you do not specify an action, Domino defaults to the ?Open action.

To require user authentication, append the Login argument to any Domino URL.

Because URLs cannot contain spaces, use the + (plus sign) as a separator. For example: http://www.mercury.com/discussion.nsf/By+Author

Separate arguments with & (ampersand). For example: http://www.mercury.com/leads.nsf/By+Salesperson?OpenView&ExpandView

Separate hierarchical names with / (slash). For example, to open a view named Docs\By Author in a database named Discussion, enter:

http://www.mercury.com/discussion.nsf/Docs/By+Author

D.1.1.1 Some specific arguments

Here is description of some specific URL arguments

$First

viewname/$First?OpenDocument - opens the first document in a view

&ParentUNID

?OpenForm&ParentUNID=123456789ABCDE - Allows you to inherit values from other documents. A great way of doing this is to have a view with the column value of...

```
"[<a href=\"/dbase.nsf/FormName?OpenForm&ParentUNID="
+@Text(@DocumentUniqueID) + "\">]" + DocumentNameField + "[</a>]"
```

Gives you the following URL in the column...

```
<a href="/dbase.nsf/FormName?OpenForm&ParentUNID=123456789ABC">Document Name</a>
```

?ReadForm or ?OpenForm

?OpenForm will open the form in edit mode, whereas ?ReadForm will open the form in preview mode, with no ability to directly edit the document.

See page 186 of the Redbook *Developing Web Applications Using Lotus Notes Designer for Domino 4.6*, SG24-2183 for more information.

D.1.1.2 URL commands for accessing documents

The following commands allow you to open a document by key, or to generate a URL to link to a document by key. Hidden design elements are hidden from the server too, so you can't use Domino URL commands to access documents in hidden views, unless you've hidden the view by surrounding its name in parentheses, rather than using the Hide tab in the Properties box. Documents to which the user does not have Access Control access to can also not be successfully referenced.

To open a document by key, create a sorted view with the sort on the first key column. Then use a URL to open the document:

Syntax

```
http://Host/Database/View/DocumentName?OpenDocument
```

Where:

View is the name of the view.

DocumentName is the string, or key, that appears in the first sorted or categorized column of the view. Use this syntax to open, edit, or delete documents and to open attached files. Domino returns the first document in the view whose column key exactly matches the DocumentName.

There may be more than one matching document; Domino always returns the first match. The key must match completely for Domino to return the document. However, the match is not case-sensitive or accent-sensitive.

Note that View can be a view Note ID, UNID, or view name. In addition, the implicit form of any of these commands will work when appropriate. (EditDocument and DeleteDocument must be explicit commands.)

Example

```
http://www.mercury.com/register.nsf/Registered+Users/Jay+Street?OpenDocume
nt
```

D.1.1.3 Some Domino URL Commands

```
?OpenServer
?OpenDatabase
?OpenNavigator
?OpenAgent
?OpenView
?OpenView&Count=150&ExpandView
?OpenView&Start=1&Count=30&CollapseView
?OpenForm
```

```
?OpenForm&ParentUNID=123456789ABCDE - Allows you to inherit values from
other documents.
?ReadForm
?OpenElement
?OpenElement&FieldElemFormat=ImageFormat
?SearchSite
?SearchView
?SaveDocument
?CreateDocument
?EditDocument
?DeleteDocument
?OpenDocument
#3 In a collapsable view takes you to the 3rd category, eg
"?OpenView&Start=1&Count=100&Expand=2#2"
&Login
&ParentUNID=
$DefaultForm
$DefaultView
$SearchForm
$Help
$File
$First
$Icon
$About
```

D.1.1.4 Domino URL commands for opening files
This technique is used for images, attachments, and OLE objects

The following commands open files and objects within a document.

?OpenElement
Use the ?OpenElement command to access file attachments, image files, and OLE objects.

Using OpenElement with file attachments

Syntax
http://Host/Database/View/Document/$File/Filename?OpenElement

Example
```
http:/
/www.mercury.com/lproducts.nsf/By+Part+Number/SN156/$File/spec.txt?OpenEle
ment
```

Note that if more than one attached file has the same name, the URL includes both the "internal" file name as well as the external name. Since the internal file name is not easily determined, make sure all attached files have unique names.

Because some browsers require that the URL end with the attached file name, Domino treats all file attachment OpenElement commands as implicit commands.

```
http://Host/Database/View/Document/$File/InternalFileName/Filename?OpenEle
ment
```

Using OpenElement with image files
Syntax

```
http://Host/Database/View/Document/FieldName/FieldOffset?OpenElement
http://Host/Database/View/Document/FieldName/FieldOffset?OpenElement&Field
ElemFormat=ImageFormat
```

Optional argument for OpenElement

```
FieldElemFormat = ImageFormat
```

Where:

ImageFormat is either **GIF** or **JPEG**. If you do not specify FieldElemFormat, Domino assumes the image file format is GIF.

Using OpenElement with OLE Objects

Syntax
http://Host/Database/View/Document/FieldName/FieldOffset/$OLEOBJINFO/FieldOffset/obj.ods?OpenElement

Note that the current URL syntax for referencing images and objects in Domino documents -- specifically the FieldOffset -- makes it impractical to create these URLs manually. As an alternative, you may paste the actual bitmap or object in place of the reference, create URL references to files stored in the file system, or attach the files to the documents.

D.1.1.5 URL commands for opening, editing, & deleting docs

The following commands manipulate documents in a database. Hidden design elements are hidden from the server too; you can't use Domino URL commands to access documents in hidden views.

?OpenDocument
Syntax

```
http://Host/Database/View/Document?OpenDocument
```

Where:

Document is any of the following:

- **DocumentKey** -- the contents of the first sorted column in the specified view.
- **DocumentUniversalID**
- **DocumentNoteID**
- **$first** -- the first document in the view

Examples

```
http://www.mercury.com/products.nsf/By+Part+Number/PC156?OpenDocument
http://www.mercury.com/leads.nsf/By+Rep/35AE8FBFA573336A852563D100741784?O
penDocument
http://www.mercury.com/leads.nsf/By+Region/0000229E?OpenDocument
```

?EditDocument
Syntax

```
http://Host/Database/View/Document?EditDocument
```

Example

```
http://www.mercury.com/products.nsf/By+Part+Number/PC156?EditDocument
```

?DeleteDocument
Syntax

```
http://Host/Database/View/Document?DeleteDocument
```

Example

```
http://www.mercury.com/products.nsf/By+Part+Number/PC156?DeleteDocument
```

?CreateDocument
The CreateDocument command is used as the POST action of an HTML form. When the user submits a form, Domino obtains the data entered in the form and creates a document.

Syntax
http://Host/Database/Form?CreateDocument

Example

```
http://www.mercury.com/products.nsf/b9815a87b36a85d9852563df004a9533?CreateDocument
```

?SaveDocument
The SaveDocument command is used as the POST action of a document being edited. Domino updates the document with the new data entered in the form.

Syntax

```
http://Host/Database/View/Document?SaveDocument
```

Example

```
http://www.mercury.com/products.nsf/a0cefa69d38ad9ed8525631b006582d0/4c95c
7c6700160e2852563df0078cfeb?SaveDocument
```

Appendix E. Naming convention used

Coding in this redbook uses the naming convention shown below (informally called the *Schramm naming convention*).

The following are the prefixes used for variables and objects, and their corresponding data types or Objects.

JavaScript Object and Data types

btn	image used as button
fn	function
form	form
i	integer
img	image
inf	inputField
is	boolean
rad	radio button
rad	radioButton
s	string
sel	select
sel	selectField
v	variant

LotusScript Object and Data types

ag	NotesAgent
dcl	NotesDocumentCollection
doc	NotesDocument
fn	NotesFunction
i	Integer
log	NotesLog
long	Long (large integer)
ndb	NotesDatabase
s	String
ses	NotesSession
sub	Sub
view	NotesView
//	comment

Appendix F. The Mapperizer PlaceBot.LSS

```
Sub Initialize

'----===oooOoo=======
    'Mapperizer
    'Created for the IBM Redbook Customizing QuickPlace
        'Written by David Wyss 2000.09.25
    'Description: cycles through the toc and finds all the docs
        'in the quickplace, relative to the toc.
        'For security reasons, this Bot does not map rooms, doing this
        'will allow people without access to a room to see that it
        'exists.
'----===oooOoo=======

'notes dims
    Dim ses As New NotesSession
    Dim ndb As NotesDatabase
    Dim docInTOC As NotesDocument
    Dim viewTOC As NotesView

'writing dims
    Dim sSysName As String
    Dim sHTMLHead As String
    Dim sHTML01 As String
    Dim sHTML02 As String
    Dim fileNum As Integer
    Dim sMainText As String
    Dim viewCurrent As NotesView
    Dim docTemp As NotesDocument
    Dim sFolderImg As String
    Dim sPageImg As String
    Dim sPageSubImg As String
    Dim sIndentImg As String
    Dim sDbPath As String
    Dim sStyleTag As String
    Dim sFolderStyleTag As String
    Dim sDetailStyleTag As String
    Dim sEndStyleTag As String
    Dim sMapDocName As String
    Dim docReport As NotesDocument
    Dim iNested As Integer
    Dim sThisSetParentUnid As String
    Dim sPrevSetParentUnid As String
    Dim sIndentConcat As String
    Dim iIndentDistance As Integer

'set up Logging
    Dim nLinkCount As Integer
    Dim logAgent As New NotesLog( ses.CurrentAgent.name )
    logAgent.LogActions = True
    Call logAgent.OpenAgentLog
    nLinkCount = 0
    Call logAgent.LogAction("Mapperizer: IBM Redbook Example" )
    Call logAgent.LogAction("Mapperizer: Run agent: " & ses.CurrentAgent.name )

'set up error handling turn this off during debugging
    On Error Goto lblLogError

'document related sets
    sMapDocName = "SiteMap"
    Set ndb = ses.CurrentDatabase
    Set viewCurrent = ndb.getView( "h_Index" )
    Set docReport = viewCurrent.GetDocumentByKey( sMapDocName )
    If docReport Is Nothing Then 'make sure the target doc exists,
        Call logAgent.LogAction("Mapperizer: Document " & sMapDocName & " not found" )
        Exit Sub
    End If
    Set viewTOC = ndb.getView("h_Toc")

'section for creating a url string
    Dim sFSlash As String
    Dim sBSlash As String
    Dim sDbLabel As String
    Dim iSlashPos As Integer
    sDbPath =  ndb.FilePath & "/"
    sFSlash$ = Chr(47)
    sBSlash$ = Chr(92)
    iSlashPos = Instr(1, sDbPath$, sBSlash$)
    While iSlashPos <> 0
        Mid$(sDbPath$, iSlashPos, 1) = sFSlash$
        iSlashPos = Instr(1, sDbPath$, sBSlash$)
    Wend 'sDbPath is now a string formatted for the web - with forward slashes

'writing sets
    sFolderImg = |<img src=../../h_index/| & sMapDocName & |/$FILE/Folder.gif
        border=0 width=20 height=13>|
    sPageImg = |<img src=../../h_index/| & sMapDocName & |/$FILE/Page.gif
        border=0 width=13 height=13>|
    sPageSubImg = |<img src=../../h_index/| & sMapDocName & |/$FILE/PageSub.gif
        border=0 width=29 height=13>|
    sIndentImg = |<img src=../../h_index/| & sMapDocName & |/$FILE/Indent.gif
        border=0 width=13 height=13>|
    sHeadingStyleTag = "<span class=h-mapHeading-text>"
    sStyleTag = "<span class=h-map-text>"
    sFolderStyleTag = "<span class=h-mapFolder-text>"
    sDetailStyleTag = "<span class=h-mapDetail-text>"
    sEndStyleTag = "</span>"
    sIndentConcat = ""
    sHTMLHead = |<html><head><link rel=stylesheet type=
        "text/css" href="Mapperizer.css"></head>| 'only for debug
    sHTML01 = |<!-- start -->| 'put JavaScripts here or the imported document
    sHTML02 = |<!-- end -->| 'this will be concantenated at the end

'start creating the documents HTML
    sMainText = sHTML01
    Set docInTOC = viewTOC.GetFirstDocument
    If docInTOC Is Nothing Then  'make sure the TOC is found
        Call logAgent.LogAction("Mapperizer: TOC Not Found" )
        Exit Sub
    End If
    sMainText = sMainText & sHeadingStyleTag & "All Documents in the " &
        ndb.Title & " QuickPlace" & sEndStyleTag

'start cycling through the documents in the toc
    While Not ( docInTOC Is Nothing )
        If docInTOC.h_Type(0) = "0" Then 'it is a doc
            Set docTemp = docInTOC
            If docTemp.h_Name(0) <> sMapDocName Then
                If docTemp.h_URLpointer(0) <> "" Then 'it is a link
                    sMainText = sMainText & "<br>" & "<a href=" &
                        docTemp.h_URLpointer(0) & ">"
                Else
                    sMainText = sMainText & "<br>" & "<a href=../../h_Index/" &
                        Cstr(docTemp.UniversalID) & "?OpenDocument>"
                End If
                sMainText = sMainText & sPageImg & sStyleTag & docTemp.h_Name(0) &
                    sEndStyleTag
                sMainText = sMainText & sDetailStyleTag
                sMainText = sMainText & " (Last Changed: " & Cstr(docTemp.LastModified)
                sMainText = sMainText & " Size: "
                If docTemp.Size < 1000 Then
                    sMainText = sMainText & "0"
                End If
                sMainText = sMainText & Cstr( docTemp.Size / 1000 ) & "k"
                sMainText = sMainText & ")" & sEndStyleTag & "</a>"
                nLinkCount = nLinkCount + 1
                'end standard formatting
            End If

        Elseif docInTOC.h_Type(0) = "1" Then 'it is a view
            sSysName = docInTOC.h_SystemName(0) 'get the internal name of the view
            'if it is a link to a quickplace tool page
            If ( sSysName = "h_Tailor" ) Or ( sSysName = "h_Members" ) Then
                'do not do anything
                'add code here if you want to map these docs as well
            Else
                iNested = 0
                If docInTOC.h_FolderStyle(0) = "5" Then 'it is a response folder
                    iNested = 1
                End If
                Set viewCurrent = ndb.getView( sSysName )
                sMainText = sMainText & "<br><a href=../../h_Index/" &
                    Cstr(docInTOC.UniversalID) & "?OpenDocument>"
                sMainText = sMainText & Chr(13) & Chr(9) &
                    sFolderImg
                sMainText = sMainText & sFolderStyleTag & docInTOC.h_Name(0) &
                    sEndStyleTag & "</a>"
                Set docTemp = viewCurrent.GetFirstDocument
                While Not (docTemp Is Nothing)
                    If docTemp.h_Name(0) <> sMapDocName Then
                        sMainText = sMainText & "<br>"
                        If docTemp.IsResponse And iNested Then
                            iIndentDistance = 13
                            'End If
                            sIndentConcat = "<img src=blank.gif width=" &
                            Cstr(iIndentDistance) & " height=1 border=0>"
                            sPrevSetParentUnid = sThisSetParentUnid
                        Else
                            iIndentDistance = 0
                            sIndentConcat = ""
                        End If
                        sMainText = sMainText & sIndentConcat
                        If docTemp.h_URLpointer(0) <> "" Then  'it is a link
                            sMainText = sMainText & "<a href=" & docTemp.h_URLpointer(0) &
                                ">"
                        Else
                            sMainText = sMainText & "<a href=../../h_Index/" &
                                Cstr(docTemp.UniversalID) & "?OpenDocument>"
                        End If
                        sMainText = sMainText & sPageSubImg & sStyleTag & docTemp.h_Name(0)
                            & sEndStyleTag
                        sMainText = sMainText & sDetailStyleTag
                        sMainText = sMainText & " (Last Changed: " &
                            Cstr(docTemp.LastModified)
                        sMainText = sMainText & " Size: "
                        If docTemp.Size < 1000 Then
                            sMainText = sMainText & "0"
```

```
                  End If
                  sMainText = sMainText & Cstr( docTemp.Size / 1000 ) & "k"
                  sMainText = sMainText & ")" & sEndStyleTag & "</a>"

                  nLinkCount = nLinkCount + 1
               End If
               Set docTemp = viewCurrent.GetNextDocument( docTemp )
            Wend
         End If
      Else
         'it must be some other sort of link, such as a room (h_Type = "3")
      End If
      Set docInTOC = viewTOC.GetNextDocument( docInTOC )
      sMainText = sMainText
   Wend

'do log handling
   Call logAgent.LogAction("Agent Mapped " & Cstr( nLinkCount ) & " documents" )
   sMainText = sMainText & "<br>" & sHTML02 & "<br>"

'debug version writes to a file
   'fileNum% = Freefile()
   'Open "D:\trash\trash.htm" For Output As fileNum%
   'Print #fileNum%, sHTMLHead ; sMainText
   'Close fileNum%

'release version writes to a db
   docReport.PageBody = sMainText
   Call docReport.Save( True, True )
   Call logAgent.Close

'the end
   Exit Sub

lblLogError:
   Call logAgent.Logerror(Err, Error$)
   Resume Next

End Sub
```

Appendix G. TheXRay Agent from TheXRay.nsf

The following file is the main agent that collects data relating to QuickPlaces on the QuickPlace server, and writes it to a file.

```
%REM /*--------===oooOoo===----
    TheXRay
    Created by David Wyss 2000.09.25
        For the IBM Redbook
    Description: cycles through directory structure on a server,
        finding all files in the QuickPlace folder which are
        either QuickPlaces or rooms.
        Due to ACL concerns, statistics are retrieved from the
        servers log.nsf.
        By signing the agent with the administrators id, the
        agent can be changed to also look into the QuickPlaces.
    Possible future changes:
        Display results based on a skin file
%END REM----===oooOoo===----

Option Public
Option Declare
Dim g_ses As NotesSession
Dim g_ndbServerLog As NotesDatabase
Dim g_vwLogDbUsage As NotesView
Dim g_dclLogEntry As NotesDocumentCollection
Dim g_sHTML As String
Dim g_sDbSizeText As String
Dim g_sDbManager As String
Dim g_sDbMonthUsesText As String
Dim g_sDbAgeText As String
Dim g_longServerSize As Long
Dim g_sURLTemp As String
Dim g_sServerName As String
Dim g_longServerSizeDisplayUnits As Long
Dim g_longMonthUsesDisplayUnits As Long
Dim g_sFolderName As String
Dim g_iRoomCount As Integer
Dim g_iPlaceCount As Integer

Sub Initialize

'//declare local variables
    Dim ndbThis As NotesDatabase
    Dim docReport As NotesDocument
    Dim dtQuery As New NotesDateTime( "" )
    Dim vwSizeGraph As NotesView
    Dim sQuery As String
    Dim sHTMLHead As String
    Dim sHTMLTitle As String
    Dim sNdbRoomSize As String
    Dim iFileNum As Integer
    Dim iDebug As Integer

'//set global variables
    Set g_ses = New NotesSession
    g_sServerName = "SvrOrca1/Orca"
    Set g_ndbServerLog = New NotesDatabase( g_sServerName , "log.nsf")
    Set g_vwLogDbUsage = g_ndbServerLog.GetView( "DatabaseUsage" )
    sQuery = |Form = "Activity" & @left( @lowercase( Pathname ) ; "\\" ) = "quickplace"|_
    & | & @rightback( @lowercase( Pathname ) ; "\\" ) != "search.nsf"|
'//GetAllDocumentsByKey wouldn't work
    Set g_dclLogEntry = g_ndbServerLog.Search( sQuery , dtQuery , 0 )
    g_iPlaceCount = 0
    g_iRoomCount = 0
    g_longServerSize = 0
    g_longMonthUsesDisplayUnits = 10
    g_longServerSizeDisplayUnits = 100000

    Print( |started agent | & g_ses.CurrentAgent.name)

'//build html header
    sHTMLHead = |<html>
    <head>
        <link rel=stylesheet type="text/css" href="../StyleSheet.css">
    </head>
    <body class=m-page-bg>
    <table width=100% border=0>|

'//initialize html body string
    g_sHTML = ||

'//set up error handling turn this off during debugging
    On Error Goto lblLogError
    Dim logAgent As New NotesLog( g_ses.CurrentAgent.name )
    logAgent.LogActions = True
    Call logAgent.OpenAgentLog
    Call logAgent.LogAction("ServerMap: IBM Redbook Example" )

'//start Place and Room reporting
    Call subGetPlace

'//build html report title
    sHTMLTitle = |<td width=26 valign=top>
        <img src=../ecblank.gif width=26 height=14 border=0>
    </td>
    <td width=216 valign=top class=m-head-text>
        <img src=../ecblank.gif width=216 height=20 border=0><br><a
href="../../TheXRay.nsf">
    <img src=../logoXRay.gif width=203 height=71 border=0 alt="Click here to see the
overview"></a>
    </td>
    <td width=100% valign=top class=m-head-text>All QuickPlaces | & Cstr(Now()) & |<img
src=../ecblank.gif width=1 height=45>
    <br>|
    sHTMLTitle = sHTMLTitle & |<span class=m-subHead-text>There are | & Cstr( Round(
g_longServerSize / 1000000 , 3 ) ) & | Mbs of QuickPlaces<br>
    There are | & Cstr(g_iPlaceCount ) & | QuickPlaces on server |
    sHTMLTitle = sHTMLTitle & g_sServerName & | and | & Cstr(g_iRoomCount)  & |
rooms</span>
    <br><a href="../../TheXRay.nsf" class=m-darkLink-text>Click here to return to Main
Menu</a></td></tr>|
    sHTMLTitle = sHTMLTitle & |</table><table width=100% border=0>|

'//combine html header, report title and body
    g_sHTML = sHTMLHead & sHTMLTitle & g_sHTML & "</table><br><br>"

'//output report
    iDebug = 0
    If iDebug Then
        'debug version writes to an HTM file
        iFileNum% = Freefile()
        Open "D:\TheXRay\TheXRay.htm" For Output As iFileNum%
        Print #iFileNum%, g_sHTML
        Close iFileNum%
    Else
        'write to a notes document in the current db
        Set ndbThis = g_ses.CurrentDatabase
        Set docReport = New NotesDocument( ndbThis )
        docReport.Form = "fmXRay"
        docReport.sQuickPlaceCount = Cstr(g_iPlaceCount )
        docReport.sRoomCount = Cstr( g_iRoomCount )
        docReport.sServer = g_sServerName
        docReport.sSErverSize = g_longServerSize
        docReport.PageBody = g_sHTML
        Call docReport.Save( False, False )
        Set vwSizeGraph = ndbThis.GetView("vwSizeGraph")
        Call vwSizeGraph.Refresh
    End If

    Print( |----- end ------|)

lblLogError:
    Call logAgent.Logerror(Err, Error$)
    Resume Next

End Sub
Function fnGetDbStats( sNdbTempFilepath As String ) As String
'declare variables for looking into the serverlog
    Dim docLogEntry As NotesDocument
    Dim docLogEntryNext As NotesDocument
    Dim iDbSize As String
    Dim sDbTextPre As String
    Dim sDbSizeTextMid As String
    Dim sDbMonthUsesTextMid As String
    Dim sDbAgeTextMid As String
    Dim sDbTextPost As String
    Dim sDbDaysOld As String
    Dim longTemp As Long

    Set docLogEntry = g_dclLogEntry.GetFirstDocument
    Do
        Set docLogEntryNext = g_dclLogEntry.GetNextDocument( docLogEntry )
        If Not  ( docLogEntryNext Is Nothing ) Then
            If (docLogEntry.Pathname(0) = sNdbTempFilepath ) Then
                Exit Do
            Else
                Set docLogEntry = docLogEntryNext
            End If
        Else
            Exit Do
        End If
    Loop

    If (docLogEntry Is Nothing) Then
        'Then we have not found the document
    End If

    'While ( Not docLogEntry.Pathname(0) = sNdbTempFilepath )
        'Set docLogEntry = g_dclLogEntry.GetNextDocument( docLogEntry )
    'Wend

    If docLogEntry.Pathname(0) = sNdbTempFilepath Then Call g_dclLogEntry.DeleteDocument(
docLogEntry )

    sDbTextPre = "<tr><td>&nbsp;</td><td width=200> &nbsp; &nbsp; "
    sDbSizeTextMid = "unknown</td><td>&nbsp;"
    sDbMonthUsesTextMid = "unknown</td><td>&nbsp;"
    sDbAgeTextMid = "unknown</td><td>&nbsp;"
    sDbTextPost = "</td></tr>"
    g_sDbManager = ""

    If Not (docLogEntry Is Nothing ) Then
        longTemp = docLogEntry.DiskSpace(0)
        g_longServerSize = g_longServerSize + longTemp
        sDbSizeTextMid = Cstr( longTemp / 1000 ) & "k"
        sDbSizeTextMid = sDbSizeTextMid & "</td><td><img src=../graphGreen.gif width="
        sDbSizeTextMid = sDbSizeTextMid & Cstr( Round( longTemp /
g_longServerSizeDisplayUnits , 0 ) )
        sDbSizeTextMid = sDbSizeTextMid & " height=10>"

        longTemp = docLogEntry.MonthUses(0)
        sDbMonthUsesTextMid = Cstr( longTemp )
        sDbMonthUsesTextMid = sDbMonthUsesTextMid & "</td><td><img src=../graphRed.gif
width="
        sDbMonthUsesTextMid = sDbMonthUsesTextMid & Cstr( Fix( longTemp /
g_longMonthUsesDisplayUnits ) )
        sDbMonthUsesTextMid = sDbMonthUsesTextMid & " height=10>"

        sDbDaysOld = Cstr( Clng( Now() ) - Clng( docLogENtry.Created ) + 1 )
        sDbAgeTextMid = sDbDaysOld & " days"
        sDbAgeTextMid = sDbAgeTextMid & "</td><td><img src=../graphBlue.gif width="
        sDbAgeTextMid = sDbAgeTextMid & sDbDaysOld
        sDbAgeTextMid = sDbAgeTextMid & " height=10>"
            'g_iDbRoomFound = True
            'Set docLogEntry = g_vwLogDbUsage.GetLastDocument
        'End If
        'Set docLogEntry = g_vwLogDbUsage.GetNextDocument(docLogEntry)
    'Wend
    End If

    g_sDbSizeText = sDbTextPre & "&nbsp;Size: " & sDbSizeTextMid & sDbTextPost
    g_sDbMonthUsesText = sDbTextPre & "&nbsp;Uses this month: " & sDbMonthUsesTextMid &
sDbTextPost
    g_sDbAgeText = sDbTextPre & "&nbsp;Age of database: " & sDbAgeTextMid & sDbTextPost

    'fnGetDbStats = g_sDbSizeText

End Function
Function fnFormatURL( sURL As String ) As String
    'format like this: fnFormatURL( "one\two\three" )
    Dim sFSlash As String
    Dim sBSlash As String
    Dim sDbLabel As String
    Dim iSlashPos As Integer
    sFSlash$ = Chr(47)
    sBSlash$ = Chr(92)
    iSlashPos = Instr(1, sURL, sBSlash$)
    While iSlashPos <> 0
        Mid$( sURL, iSlashPos, 1) = sFSlash$
        iSlashPos = Instr(1, sURL, sBSlash$)
    Wend 'sDbPath is now a string formatted for the web - with forward slashes
    fnFormatURL = sURL
End Function
Sub subGetRoom
    Dim dirRoom As NotesDbDirectory
    Dim ndbRoom As NotesDatabase
    Dim sRoomTemp As String
    Dim sRoomName As String
    Dim sNdbRoomFilepath As String
    Dim sTempFilepath As String
    Dim sHTMLRoomIn As String
    Dim sHTMLRoomOut As String
    Dim sHTMLAfterRoomTitle As String


    Set dirRoom = New NotesDbDirectory( g_sServerName )
    Set ndbRoom = dirRoom.GetFirstDatabase( DATABASE )
    sHTMLRoomIn = |<tr height=5>
    <td width=40 height=5><img src=../ecblank.gif width=40 height=5></td>
    <td width=200><img src=../ecblank.gif width=200 height=5></td>
    <td width=100%><img src=../ecblank.gif width=200 height=5></td>
</tr>
<tr><td width=40>&nbsp;</td>
<td class=m-room-text>&nbsp;&nbsp;&nbsp;<img src=../icnRoom.gif align=absmiddle width=17
height=14 border=0>&nbsp;|
    sHTMLRoomOut =  |</td></tr>|
    sHTMLAfterRoomTitle = |</td><td width=100%>&nbsp;<!-- aftertitle 1 --></td></tr>|

    While Not ( ndbRoom Is Nothing )
        sNdbRoomFilepath = ndbRoom.FilePath
        sTempFilepath = |quickplace| & g_sFolderName & |pagelib|
        If Instr(1, Lcase( sNdbRoomFilepath ), sTempFilepath) Then
            sRoomName = ndbRoom.Title
            sRoomTemp = sNdbRoomFilepath
            g_sURLTemp = "<a href=/" & fnFormatURL( sRoomTemp ) & " class=m-place-text>"
            sRoomTemp = sHTMLRoomIn & g_sURLTemp & sRoomName & "</a>" &
sHTMLAfterRoomTitle
            'If Instr( 1 , sNdbRoomFilepath , "PageLib") Then Print( sNdbRoomFilepath & "
- Roomfilepath, should be in next comparison")
            Call fnGetDbStats( sNdbRoomFilepath )
            sRoomTemp = sRoomTemp & g_sDbSizeText & g_sDbMonthUsesText & g_sDbAgeText
            g_sHTML = g_sHTML + sRoomTemp
            sRoomTemp = ""
            g_iRoomCount = g_iRoomCount + 1
        End If
        Set ndbRoom = dirRoom.GetNextDatabase
    Wend
End Sub
Sub subGetPlace
    Dim dirPlace As NotesDbDirectory
    Dim ndbPlace As NotesDatabase
    Dim sPlaceTemp As String
    Dim sPlaceName As String
    Dim sNdbPlaceFilepath As String
    Dim sHTMLPlaceIn As String
    Dim sHTMLPlaceOut As String
    Dim sHTMLAfterPlaceTitle As String
    Dim iFolderNameLength As Integer

    sHTMLPlaceIn  = |<tr height=5>
        <td width=40 height=5><img src=../ecblank.gif width=40 height=5></td>
        <td width=200><img src=../ecblank.gif width=200 height=5></td>
        <td width=100%><img src=../ecblank.gif width=200 height=5></td>
    </tr>
    <tr><td width=40>&nbsp;</td>
        <td class=m-place-text><img src=../icnPlace.gif align=absmiddle width=17
height=14 border=0>&nbsp;|
    sHTMLPlaceOut = |</td></tr>|
    sHTMLAfterPlaceTitle = |</td><td width=100%>&nbsp;<!-- aftertitle 1 --></td></tr>|


    Set dirPlace = New NotesDbDirectory( g_sServerName )
    Set ndbPlace = dirPlace.GetFirstDatabase( DATABASE )

    While Not ( ndbPlace Is Nothing )
        sNdbPlaceFilepath = ndbPlace.FilePath
        If Instr(1, Lcase( sNdbPlaceFilepath ), |quickplace\| ) Then
            If ndbPlace.FileName = "Main.nsf" Then
                g_sURLTemp = "<a href=/" & fnFormatURL( sNdbPlaceFilepath ) & "
class=m-place-text>"
                sPlaceName = ndbPlace.Title
                sPlaceTemp = sHTMLPlaceIn & g_sURLTemp & sPlaceName & "</a>" &
sHTMLAfterPlaceTitle
                g_iPlaceCount = g_iPlaceCount + 1
                sNdbPlaceFilepath = ndbPlace.FilePath
                iFolderNameLength = Len( sNdbPlaceFilepath ) - 19
                g_sFolderName = Lcase( Mid( sNdbPlaceFilepath , 11 , iFolderNameLength ) )
& |\|
                Call fnGetDbStats( sNdbPlaceFilepath )
                sPlaceTemp = sPlaceTemp & g_sDbSizeText & g_sDbMonthUsesText &
g_sDbAgeText
                g_sHTML = g_sHTML + sPlaceTemp
                sPlaceTemp = ""

                'now we have a main, get its rooms
                Call subGetRoom

                Set ndbPlace = dirPlace.GetNextDatabase
            End If
        End If
        Set ndbPlace = dirPlace.GetNextDatabase
    Wend
End Sub
```

Appendix H. EditFavorites.HTM page

The EditFavorites page makes it possible to edit the links you have created via the Add this page item in the Millennia Theme. It is described in detail in Chapter 4, “Creating Themes” on page 47.

```
<HTML>
<HEAD>
<TITLE></TITLE>
<META name="description" content="">
<META name="keywords" content="">
</HEAD>
<BODY BGCOLOR=FFFFFF TEXT=000000 LINK=0000FF VLINK=800080>
<!-- everything ABOVE this line will be removed by quickPlace -->
<!-- /*--------=oooOoo===----
    'EditFavorites.htm Favorites Manager
    'Created:
        'By David (-the bush kangaroo-) Wyss and Viktor (-Vicki-) Krantz
        '2000.09.25
        'For the IBM Redbook
    'Description: This file uses cookies to create a favorites
        'list. In order to make this more flexible to the user
        'we put in the <select> and some funky buttons.
        'We have also put in a Limerick image which uses a URL
        'to create an image on the server.
        'This page is not as complex as it looks, it is so long
        'mostly because of the Limerick preparation.
    '----===oooOoo===----
*/ -->
<!-- // start the page off by creating a series of images on the page to force
       QuickPlace to upload them -->
<img name=ReorderDown2 src=ReorderDown.gif width="1" height="1" border=0>
<img name=ReorderDownHL src="ReorderDownHiLite.gif" width="1" height="1" border=0>
<img name=ReorderUp2 src=ReorderUp.gif width="1" height="1" border=0>
<img name=ReorderUpHL src="ReorderUpHiLite.gif" width="1" height="1" border=0>
<img name=Delete2 src=Delete.gif width="1" height="1" border=0>
<img name=DeleteHL src="DeleteHiLite.gif" width="1" height="1" border=0>
<script language=JavaScript>
function fnMoveUpOrDown(sDirection) {
    //main function for moving the links up and down
    //initialize the string variables and set them to blank
    //note that objects in this file have the following prefixes:
    //  s = string
    //  i = integer
    //  fn= function
    //  v = variant
    //  inf= inputField
    //  sel= select
    //  rad= radio button
    //  t = thing
    //  form= form
    var sSourceText = sSourceValue = sDestText = sDestValue = "";
    //initialize the initger variables and set them to zero
    var iSourceIndex = iDestIndex = 0;
    //initialize and set a variable to contain the field we will be manipulating
    var vSelectedObject = document.forms.formFavorites.selMain;
    //initialize a variable and put the number of the selected line in it
    //remember that the numbering starts at zero
    var iSourceIndex = vSelectedObject.selectedIndex;

    //switch to up or down mode depending on the sDirection variable
    //if the iDownDirection is true, set variables to move the selected line down
    switch (sDirection) {
        case 'down':
            if (iSourceIndex +1 == vSelectedObject.length) return;
            iDestIndex = iSourceIndex + 1;
            break;
        case 'up':
            if (iSourceIndex < 1) return;
            iDestIndex = iSourceIndex - 1;
            break;
    }

    //the following statements reference the vSelectedObject.
    //to make scripting shorter they are contained in a with
    //statement. This appends vSelectedObject to each
    //argument in the bounds of the with statement.
    with (vSelectedObject) {
        //the following set of arguments perform a sort of
        //swapping mechanism.
        //first of all the selected item and the one it going to
        //replace are put into variables, then they are swapped over.
        sSourceText = options[iSourceIndex].text;
        sSourceValue = options[iSourceIndex].value;
        sDestText  = options[iDestIndex].text;
        sDestValue = options[iDestIndex].value;
        options[iDestIndex].text = sSourceText;
        options[iDestIndex].value = sSourceValue;
        options[iSourceIndex].text = sDestText;
        options[iSourceIndex].value = sDestValue;
        selectedIndex = iDestIndex;
    }
    fnChangeCookies( 'none' )
}

function fnReMoveSel( sAction  ){
    //this function removes items from the select, and from the
    //cookies.
    if( document.forms.formFavorites.selMain.length == 0) return;
    if ( sAction == 'delete'){
        var iConfirm = confirm( 'Are you sure you want to delete this Favorite?' );
        if ( !iConfirm ) { return; };
    }
    var vSel = document.forms.formFavorites.selMain;
    vSel.options[ vSel.selectedIndex ] = null;
    vSel.selectedIndex = 0;
    fnChangeCookies( 'delete' )
};

function fnGetCookie( sCookieName ) {
    // return the named cookie.s value
    if( document.cookie){
        index = 0;
        index = document.cookie.indexOf(sCookieName);
        if(index != -1){
            countbegin = ( document.cookie.indexOf("=" , index) + 1);
            countend =     document.cookie.indexOf(";" , index);
            if (countend == -1){
                countend = document.cookie.length;
            };
            return document.cookie.substring(countbegin , countend);
        } else { return '' }; // otherwise may return undefined
    }  else { return '' } ;
};
```

```
//declare the variable to be used in the whole page
//be careful if you write new functions, you may need to
//reuse this line to refresh the current cookie list.
var sExistingList = fnGetCookie( 'cookFavorites' );

function fnSetCookie( sCookieName, sCookieValue, sCookieDays ) {
    // call with these arguments fnSetCookie( name-of-the-cookie, text-value ,
[days-to-keep] )
    // Set the daystokeep to 0 to destroy the cookie.
    if ( sCookieDays == null) { //user has not set a date
        sCookieDaysString = '' ;
    } else {
        var dtCookieDays = new Date();
        var today = new Date() ;
        dtCookieDays.setTime(today.getTime() + 1000*60*60*24* sCookieDays + 100);
        sCookieDaysString = "; expires=" + dtCookieDays.toGMTString();
    };
    document.cookie = sCookieName + "=" + sCookieValue + sCookieDaysString + ' ; path=/
;';
};

function fnAppendCookie( sCookieName, sCookieValue, sCookieDays ) {
    //adds a new cookie to the existing cookie list
    var sLabel = prompt("What would you like to call this link?", h_Name );
    if ( sLabel != null && sLabel != ''){
        var sNewListItem = sLabel + '$$$' + document.location + '&&&'
        fnSetCookie( sCookieName , sExistingList + sNewListItem , sCookieDays ) ;
    }
};

//declare this variable here so that it can be used in all funcions
var iCookiesSaved;

function fnWriteFavSelect(){
    //this function writes the select into the file, using cookies
    var sExisingList = fnGetCookie( 'cookFavorites' )
    var asExistingList = sExisingList.split('&&&');
    var sCurrentCookie = ''
    if ( ( sExistingList.length > 0 ) && ( sExistingList.indexOf( '$$$' ) != -1 )) {
        for ( i = 0 ; i < asExistingList.length ; i++ ){
            sCurrentCookie = asExistingList[i];
            iEndOfLabel = sCurrentCookie.indexOf( '$$$' );
            if ( iEndOfLabel != -1 ) {
                sCurrentCookieLabel = sCurrentCookie.substring( 0 , iEndOfLabel );
                document.write( '<option value=' + i + '>' + sCurrentCookieLabel +
'</option>' )
            }
        }
        iCookiesSaved = 1
    } else {
        document.write('<option value=0>No saved favorites</option>');
        iCookiesSaved = 0
    }
}

function fnKillAllCookies(){
    //not in use, but this is a handy but deadly function,
    //especially when debuggerizing.
    if( confirm("Are you sure you want to delete all your favorites?") ){
        fnSetCookie('cookFavorites' , 'null' , 0 );
    }
};
```

```
function fnChangeCookies( sAction ){
    //this function merges newcookies, and leaves out the unwanted cookies
    if ( !iCookiesSaved ){ return };
    var sList = "";
    var sTemp = fnGetCookie( 'cookFavorites' );
    var sOldCookies = sTemp.split( '&&&' )
    var sNewCookies = '';
    var iSelectedCookie;
    var iCookiesToSave = document.forms.formFavorites.selMain.length

    for (i = 0 ; i < iCookiesToSave ; i++){
        iSelectedCookie = parseInt( document.forms.formFavorites.selMain.selectedIndex );
        sList = eval( 'document.forms.formFavorites.selMain.options[' + i + '].value' )
        sNewCookies += eval( 'sOldCookies[ ' + sList + ' ]') + '&&&' ;
    }
    fnSetCookie( 'cookFavorites' , sNewCookies , 365 )
}

// the following functions are for writing the title
function fnPrintName(){
    //this function sends a URL to the server, the server creates
    //a gif image and sends back the file.  This form of technology
    //uses the Domino Graphics Server.
    //the first thing we do here is check that the REMOTE_USER variable
    //has been declared. If it has not been declared, it means the file is not
    //being accessed whilst being run from a QuickPlace. In this case the
    //Graphics Server will not engage and we will see a blank image.
    if( typeof REMOTE_USER == 'undefined'){
        var sImgSRC = '<b>Edit Favorites</b><br><br>'
    } else {
        var sImgSRC = '<img src=../../Main.nsf?';
        sImgSRC += 'GetImage&TextString=Favorites%20For%20';
        sImgSRC += escape( fnStringFormatt( REMOTE_USER ) );
        sImgSRC += '&EffectType=Shadow&FontBold=1&FontItalic=1';
        sImgSRC += '&FontName=Verdana&FontPointSize=14';
        sImgSRC += '&fontColor=80CCFF&Opacity=100&BkColor=FFFFFF';
        sImgSRC += '&blurFactor=5&shadowColor=80FFFF';
        sImgSRC += '&ShadowXOffset=2&ShadowYOffset=4 border=0>';
    //  sImgSRC += '&FontBold=1&FontName=Verdana&FontPointSize=14&fontColor=20A8FF';
    //  sImgSRC += '&effectType=Shadow&blurFactor=8';
    //  sImgSRC += '&shadowXOffset=0&shadowYOffset=0&shadowColor=20FFEE\">';
    //  sImgSRC += REMOTE_USER;
    }
    document.write( sImgSRC )
};
function fnStringFormatt(sTxt) {
    //this function returns the common name of a person
    //from a full Domino canonical name.
    iTxtStart = sTxt.indexOf('=');
    iTxtStart++;
    iTxtEnd = sTxt.indexOf('/');
    if(iTxtEnd == -1) iTxtEnd = sTxt.length;
    sTxt = sTxt.substr(iTxtStart,iTxtEnd - iTxtStart);
    return sTxt;
};
// end functions are for writing the title

</script>

<table width="100%" border="0" cellpadding="0" cellspacing="0">
    <tr>
        <td width="90" valign=bottom bgcolor=FFFFFF>
```

```
            <img src="ecblank.gif" border=0 height="1" width="90"><br><img
src="icnFavorites.gif" width="73" height="59">
        </td>
        <td width=100% valign=bottom bgcolor=FFFFFF>
            <script language=JavaScript>fnPrintName();</script>
        </td>
    </tr>
    <tr>
        <td>&nbsp;&nbsp;</td>
        <td>&nbsp;&nbsp;</td>
    </tr>
    <tr>
        <td colspan=2><br>
            &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<img src="icnBulletTurquise.gif" width="16"
height="16"> Select a favorite and click the up or down arrow to move it.<br>
            &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<img src="icnBulletTurquise.gif" width="16"
height="16"> To delete a favorite, select the favorite then click on the red delete
icon.<br>
        </td>
    </tr>
    <tr>
        <td>
            <img src="ecblank.gif" border=0 height="1" width="90"><br>
        </td>
        <td><br>
            </form>
            <form name="formFavorites">
                <table border=0>
                    <tr>
                        <td width=120>
                            <select name="selMain" colwidth=40 size="7">
                            <!-- This is where the main select is created -->
                            <script>fnWriteFavSelect()</script></select>
                            <script>document.forms.formFavorites.selMain.selectedIndex =
0;</script>
                        </td>
                        <td>
                            <a href="javascript:void(0)"
onClick="javascript:fnMoveUpOrDown('up');return false;"
onmouseOver="ReorderUp.src=ReorderUpHL.src"
onmouseOut="ReorderUp.src=ReorderUp2.src"><img name=ReorderUp alt="Click here to move the
selected favorite up" src=ReorderUp.gif width=23 height=24 border=0></a><p>
                            <a href="javascript:void(0)"
onClick="javascript:fnMoveUpOrDown('down'); return false;"
onmouseOver="ReorderDown.src=ReorderDownHL.src"
onmouseOut="ReorderDown.src=ReorderDown2.src"><img name=ReorderDown alt="Click here to
move the selected favorite down" src=ReorderDown.gif width=23 height=24 border=0></a><p>
                            <a href="javascript:void(0)"
onClick="javascript:fnReMoveSel('delete'); return false;"
onmouseOver="Delete.src=DeleteHL.src" onmouseOut="Delete.src=Delete2.src"><img
name=Delete alt="Click here to delete the selected favorite" src=Delete.gif width=23
height=23 border=0></a>
                        </td>
                    </tr>
                </table>
                <p><!-- These are really handy for debugging the page if you make changes
                    <input type=button value="Save Changes" onClick="fnChangeCookies(
'save' )"> &nbsp;
                <p><input type=button value=AppendCookies onClick="fnAppendCookie(
'cookFavorites' , 'save Two$$$value&&&' , 1 )">
                <p><input type=button value=ShowCookies
onClick="alert('-'+sExistingList+'-')"> -->
            </form>
            <br>&nbsp;
        </td>
    </tr>
    <tr>
        <td colspan=2>
            &nbsp;&nbsp;<br>
        </td>
    </tr>
</table>
</BODY>
</HTML>
```

H.1 The Favorites component

The Favorites component is the scripting required to enable users to add new favorites. This coding is included in the Themes layout, but is also included separately here so that you can add it to your own layouts.

```
<HTML>
<HEAD>
<TITLE>Display Cookies Page</TITLE>
<META name="description" content="">
<META name="keywords" content="">
<link rel=stylesheet type="text/css" href="StyleSheet.css">
</HEAD>
<BODY BGCOLOR=FFFFFF TEXT=000000 LINK=0000FF VLINK=800080>

<script language=JavaScript>
    NS = (document.layers)? true:false
    IE = (document.all)? true:false
    var sURL = '../../h_Index/Edit+Favorites?OpenDocument'
    if (typeof h_Name == 'undefined' ){
        var h_Name = document.title
        sURL = 'EditFavorites10.htm'
    }
    function fnGetCookie( sCookieName ) {
        // return the cookie value
        if( document.cookie){
            index = 0;
            index = document.cookie.indexOf(sCookieName);
            if(index != -1){
                countbegin = ( document.cookie.indexOf("=" , index) + 1);
                countend =     document.cookie.indexOf(";" , index);
                if (countend == -1){
                    countend = document.cookie.length;
                };
                return document.cookie.substring(countbegin , countend);
            } else { return '' }; // otherwise may return undefined
        }  else { return '' } ;
    };

    function fnSetCookie( sCookieName, sCookieValue, sCookieDays ) {
        // call with these arguments fnSetCookie( name-of-the-cookie, text-value ,
[days-to-keep] )
        // Set the daystokeep to 0 to destroy the cookie.
        if ( sCookieDays == null) { //user has not set a date
            sCookieDaysString = '' ;
        } else {
            var dtCookieDays = new Date();
            var today = new Date() ;
            dtCookieDays.setTime(today.getTime() + 1000*60*60*24* sCookieDays + 100);
            sCookieDaysString = "; expires=" + dtCookieDays.toGMTString();
        };
        document.cookie = sCookieName + "=" + sCookieValue + sCookieDaysString + ' ;
path=/ ;';
    };

    var sExistingList = fnGetCookie( 'cookFavorites' );

    function fnAppendCookie( sCookieName, sCookieValue, sCookieDays ) {
        var sLabel = prompt("What would you like to call this link?" , h_Name );
        var sLocationURL = '';
        if ( sLabel != null && sLabel != ''){
            sLocationURL = document.location;
            var sNewListItem = sLabel + '$$$' + sLocationURL + '&&&'
            fnSetCookie( sCookieName , sExistingList + sNewListItem , sCookieDays ) ;
        }
    };
    function fnDisplayFavorites(){
        var asExistingList = sExistingList.split('&&&');
        var sCurrentCookie = ''
        var sCurrentCookieURL = '';
        document.write( '<span class=h-toolMenuGray-text>Favorites</span>' )
        if ( ( sExistingList.length > 0 ) && ( sExistingList.indexOf( '$$$' ) != -1 )) {
            for ( i = 0 ; i < asExistingList.length ; i++ ){
                sCurrentCookie = asExistingList[i];
                iEndOfLabel = sCurrentCookie.indexOf( '$$$' );
                if ( iEndOfLabel != -1 ) {
                    sCurrentCookieLabel = sCurrentCookie.substring( 0 , iEndOfLabel );
                    sCurrentCookieURL = sCurrentCookie.substring( iEndOfLabel + 3 ,
sCurrentCookie.length );
                    document.write( '<br><a href=\"' + sCurrentCookieURL + '\"
class=h-toolMenu-text alt=\"Click here to go to your favorite\">' + sCurrentCookieLabel +
'</a>' )
                }
            }
        } else {
            document.write('');
        }
    }

    function fnDisplayAdd(){
        document.write('<br><a href=\"javascript:void(0)\" class=h-toolMenu-text
onClick=\"fnAppendCookie(\'cookFavorites\', document.location , 30 )\">Add this
page...</a>')
    }

    function fnDisplayEdit(){
        document.write('<br><a href=\"'+sURL+'\" target=_top class=h-toolMenu-text>Edit
Favorites...</a>')
    }
</script>
<body>
line before<br><br>
<script>
    fnDisplayFavorites();
    fnDisplayAdd();
    fnDisplayEdit();
</script>
<br><br>line after<br><br>

</BODY>
```

```
</HTML>
```

Appendix I. ReplaceSubstring_ function

This appendix lists the source code for the function ReplaceSubstring_ referred to in 10.4.2, “Creating Places from the Domino Workflow environment” on page 303. We used it to replace the white space (or blank) character with the plus sign (+) character because blank space characters are not allowed in URLs.

```
Function ReplaceSubstring_( arg_st$, arg_stFind$, arg_stReplace$ ) As
Variant
   Dim intStart%, intStartSearch%, intLenReplace%, intLenFind%,
intReplaceCount%
   intStartSearch% = 1
   If arg_st$ = "" Then
      Exit Function
   End If
   Do While True
      intStart% = Instr( intStartSearch%, arg_st$, arg_stFind$ )
      If intStart% = 0 Then
         Exit Do
      End If
      intLenReplace% = Len( arg_stReplace$ )
      intLenFind% = Len( arg_stFind$ )
      If ( intStart% + intLenFind% - 1 ) > Len( arg_st$ ) Then '--- avoid
illegal function call
         arg_st$ = Left( arg_st$, intStart% - 1 ) + arg_stReplace
         intReplaceCount% = intReplaceCount% + 1
         Exit Do
      Else
         arg_st$ = Left( arg_st$, intStart% - 1 ) + arg_stReplace$ + Right(
arg_st$, Len( arg_st$ ) - ( intStart% + intLenFind% - 1 ) )
         intStartSearch% = intStart% + intLenReplace%
         intReplaceCount% = intReplaceCount% + 1
      End If
   Loop
   ReplaceSubstring_ = intReplaceCount%

End Function
```

Appendix J. The extended QuickPlace object model map

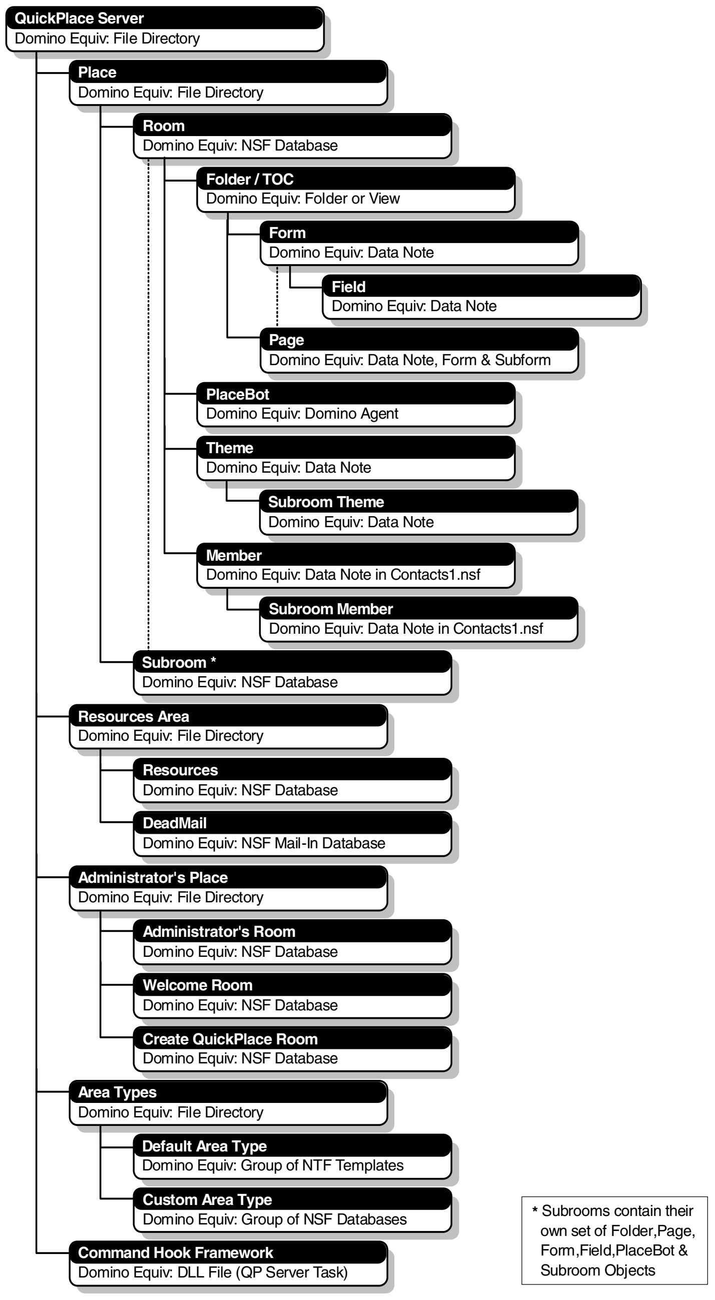

QuickPlace Server
Domino Equiv: File Directory
Place
Domino Equiv: File Directory
Room
Domino Equiv: NSF Database
Folder / TOC
Domino Equiv: Folder or View
Form
Domino Equiv: Data Note
Field
Domino Equiv: Data Note
Page
Domino Equiv: Data Note, Form & Subform
PlaceBot
Domino Equiv: Domino Agent
Theme
Domino Equiv: Data Note
Subroom Theme
Domino Equiv: Data Note
Member
Domino Equiv: Data Note in Contacts1.nsf
Subroom Member
Domino Equiv: Data Note in Contacts1.nsf
Subroom *
Domino Equiv: NSF Database
Resources Area
Domino Equiv: File Directory
Resources
Domino Equiv: NSF Database
DeadMail
Domino Equiv: NSF Mail-In Database
Administrator's Place
Domino Equiv: File Directory
Administrator's Room
Domino Equiv: NSF Database
Welcome Room
Domino Equiv: NSF Database
Create QuickPlace Room
Domino Equiv: NSF Database
Area Types
Domino Equiv: File Directory
Default Area Type
Domino Equiv: Group of NTF Templates
Custom Area Type
Domino Equiv: Group of NSF Databases
Command Hook Framework
Domino Equiv: DLL File (QP Server Task)
* Subrooms contain their own set of Folder,Page, Form,Field,PlaceBot & Subroom Objects

Figure 140. Map of the Extended QuickPlace Object Model

Appendix K. Zen and the art of QuickPlace customization

The great path has no gates,

Thousands of roads enter it.

When one passes through this gateless gate,

He walks freely between heaven and earth.

- Mu-mon 1228

In this redbook we've been walking the great path of QuickPlace customization, mostly on the technical side of the road. However, technology is only an enabler. Successful solutions depend on a solid technical foundation as QuickPlace, but also require an intimate understanding of the people and processes involved. This appendix contains considerations about what QuickPlace is, why you would want to customize it, and how to deliver Turnkey solutions from a view that involves more than the purely technical aspect.

K.1 Introduction to QuickPlace

"The most powerful tools are always simple in concept, but they often must be applied with some sophistication."—Alan Cooper, "Father of Visual Basic"

In this section we discuss QuickPlace as a collaborative solution geared to the needs of project-based teams and embedded with powerful functions typical of each of the major types of groupware: Custom Places, integrated messaging, threaded discussion, workflow automation, document management, and shared calendars.

However, QuickPlace is more platform than product. We need to go deeper to understand QuickPlace, both as a platform for distributed content management and as a platform for distributed application development. Therefore, we discuss the importance of using interactive design to drive customization, and also offer a glimpse of the ecology of innovation and reuse that QuickPlace *the platform* can support.

Finally, lest all of your hard work to extend, integrate, and reuse QuickPlace result in idle demonstrations of technical prowess, you need to understand issues of culture and organizational preparedness necessary for positive outcomes with QuickPlace (or any groupware solution).

K.1.1 What is QuickPlace

QuickPlace is groupware that empowers users to create an online collaborative workspace. *Collaboration* is the act of shared creation or shared discovery. QuickPlace provides “instant collaboration”. It lets you easily create shared workspaces on the Internet or on an intranet, without any specialized technical knowledge.

The challenge of collaboration is to address user needs of the type expressed in the following statements:

- I need to find people who can help me do my job.
- I need to let others know about this issue, and my opinion about it.
- I need to know who handled this issue the last time and what they did.
- I need a single place to organize status for things I'm involved in.
- I need a way to track the value of this information now and in the future.

K.1.1.1 Online collaborative workspace

With QuickPlace, anyone with a Web browser can effortlessly create online meeting and collaborative applications to work together, as well as share ideas, ask questions, find answers, and form communities. We would like to electronically see, hear, and interact with a geographically disconnected person or group of people as though they were not separated. We want to exchange information and create knowledge, whether in real time (synchronous collaboration), or episodically, over a period of time (asynchronous collaboration).

K.1.1.2 Groupware

Groupware is technology designed to facilitate the work of groups. Although there are many classes of groupware, we are interested in the types of groupware that rely on computer networks to allow groups to communicate, cooperate, coordinate, solve problems, compete, or negotiate (see Figure 141 on page 405). Our field of study is called Computer-Supported Cooperative Work (CSCW), which examines the design, adoption, and use of groupware. It is, by necessity, a multidisciplinary field and typically attracts those interested in software design and social and organizational behavior, including business people, computer scientists, organizational psychologists, communications researchers, and anthropologists, among other specialties.

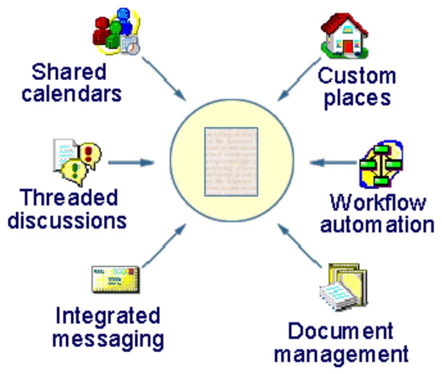


Figure 141. Types of groupware applications that are the focus of CSCW

K.1.2 The major benefits of using QuickPlace

QuickPlace provides a rich set of features closely linked to the goals of people collaborating in teams.

It is easy to use and customize for quick and easy teamwork.

- It is inexpensive to license, easy to install, and makes it easy to create Places.
- It is self-regulating—it allows companies to become more distributed and workforces to become more flexible.

You can develop and reuse tools and content with PlaceTypes

- It help you maintain or improve corporate standards.
- You can reduce team setup time.
- It ensures access to available resources.

It ensures secure collaboration, asynchronous and synchronous.

- Discussions track issues as they develop over time.
- E-mail notification alerts users that something new requires their attention.
- Real-time chat provides immediate communication in context.

It is ready to integrate into enterprise applications

- Built-in workflow and document management integrate into enterprise platforms.

- It is URL-driven, and ready to integrate into directories.
- You can edit MS Office documents in their native format.
- You can send e-mail directly into QuickPlace, regardless of platform.

K.1.3 People need a virtual meeting place to share things

Knowledge—more importantly, *people* who have knowledge—has become the defining asset of the enterprise. Knowledge is discovered, more often than not, in unstructured exchanges in which people come into contact with one another to create, share, or talk about *things*—content, both unstructured and structured. Teams of people work collectively, collaboratively, and securely to:

- Make decisions
- Share information
- Coordinate actions

Teams are often ad hoc, formed “just in time” to solve a particular problem, and then disbanded. (Ehrlich 160). However, as an organization grows, it becomes more challenging to bring people together to share ideas, experience and feedback. When people are separated by distance or time, they need a virtual *place* to share things—and they need it fast.

K.1.3.1 A place to store the knowledge of teams

QuickPlace revolutionizes content creation and content management. As teams create and use Places to meet, share, and organize documents, the Places evolve into repositories for lessons learned and best practices of the team. You can get new members of a team up to speed quickly by letting them go to the Place where these documents reside. Members can easily link to other content on the Web, bring content created in other desktop applications to the Place or create content directly in the place.

QuickPlace’s import options make content accessible to anyone with a Web browser, while round-trip editing allows content to be created and edited in its original format (see *The QuickPlace User’s Guide*).

In any organization, while one individual or team struggles to address an issue, it is possible that another individual or team has already addressed it or is still struggling with it. Finding a Place can mean finding an answer, if not in the things contained in the Place, then in the people associated with the Place.

QuickPlace's approach to content creation encourages sharing and reuse. Place content is created in the context of a specific project, program or event, but its reuse potential is unlimited. While QuickPlace can secure content from unauthorized use, a team can make some or all of its documents accessible to others, or preserve them for reference or reuse beyond the scope of the Place in which they were created.

Documents created in an instantiation of QuickPlace can be linked from other Places or pushed with each new instantiation, as they are identified as the preferred process for a particular kind of project or activity, corresponding to a particular kind of Place (a PlaceType). In this way, QuickPlace bundles content with processes, while allowing local work *practices* (often unstructured) to prevail.

11.7.2.2 Where you can QuickPlace

While there is no limit to the life cycle of Places, QuickPlace's integrated collaboration, document management, and process management capabilities are geared to support Places limited by the life cycle of a particular team, project or event.

You may want to use QuickPlace as the entry point to more sophisticated solutions. Applications that coordinate work processes, commonly known as *workflow*, are notoriously difficult to introduce into an organization, because they depend on business processes that are often obscure. Worse yet, while participants in the process may agree about the order and steps of the process, in *practice*, the work is anything but routine. As a requirements analyst or developer, you can ask people about their work, but even if they are among the few people who reflect on what they do, they are likely to be too engaged in what they do to tell you in detail what happens, or to step back and explain the bigger picture of the processes they support.

QuickPlace has built-in workflow options allowing you to direct documents to one or more people for approval or action before publishing and notification of other members. A developer can customize QuickPlace's internal document handling actions to support more sophisticated processes as a bridge to the adoption of an workflow solution, such as Domino WorkFlow, which supports process modeling and complex organizational relationships.

Just as a team might "graduate" to become users of more sophisticated workflow, they might eventually need more structured document management than the check-in/check-out, draft, and versioning that QuickPlace supports out of the box. As with workflow, QuickPlace's organic document management is an excellent way for users to support unstructured work practices, while learning learn how a more structured process might be implemented. As with workflow, developers can extend QuickPlace's document handling actions and even integrate them directly into an enterprise document management solution, such as Domino.Doc, while keeping the structure and metadata of the original documents intact.

K.1.4 Distributed application development platform

QuickPlace has the potential to dramatically change not only the way content is managed, but also the way applications are designed and developed. As a developer or administrator, you may find it difficult to explain how applications are developed for the same reason that the users of your applications are often at a loss to explain the work they do—it's hard to see the bigger picture when you are in the middle of it!

As groupware applications are often developed in direct consultation with the user (Ehrlich 145), you can leverage QuickPlace itself as a collaborative application development platform to support the teamwork of developers in different locations, different organizations, and with differing skill sets.

K.1.4.1 How to approach customization with QuickPlace

There's a joke in the computer software business that says, "Design is what programmers do in the twenty minutes before they start coding." There's another saying that, "The first ninety percent of your time you spend *writing* the code. The second ninety percent you spend *debugging* the code." We believe there may be a connection.

While some of today's visually-oriented development tools make it easier than ever to create complex applications quickly, creating good applications is another matter entirely. By *good* we mean a pleasure to use, while at the same time powerful in satisfying users' corporate, practical, and personal goals. We believe that good interaction design requires a process that is separate and distinct from the development process. For an understanding of interaction design, we recommend that you read *The Inmates Are Running the Asylum*, by Alan Cooper (1999, SAMS, a division of Macmillan Computer Publishing).

Cooper believes that too many developers approach design from the wrong perspective, concentrating on the tasks that a program must perform, rather than on the user's goals. (Cooper 151). The result is a program that appears to work, even as its user fails. That's not good design.

In designing *groupware*, it makes even less sense to let tasks drive the process because, unlike single-user applications, groupware is fundamentally about people's work and their relationships with one another. To uncover hidden work practices, and translate them into an application requires a multidisciplinary approach, embracing not just technology, but “social and management sciences” (Ehrlich 164).

At the start of a typical engagement to design and field a groupware application, your user-customer might say, “I don’t know what I want, but I’ll know it when I see it.” Now you have to decide whether to shut the door in the customer’s face (not a good idea), or send one or more highly-skilled developers to wander around in the wilderness on an expensive voyage of discovery. Consider a popular cartoon strip with the following sequence:

- Frame 1
  - Programmer: - (sits at his PC)
  - Manager: The Web site needs to be more webbish.
- Frame 2
  - Programmer: - (sits at his PC)
  - Manager: But not too webbish.
- Frame 3
  - Programmer: - (sits at his PC)
  - Manager: How long will that take?

This is a good illustration of what we call “Lost without a map”, or “I’ll know it when I see it.”

Ultimately, this voyage is too expensive for either user-customer or developer, because the destination, or more accurately, the final resting place, is often an application that fails to satisfy one of the user’s personal, practical, or corporate goals.

Because the user might not articulate personal, practical, and corporate goals, and because these goals must drive the design of the application, designers should form a design team, which we will call the *QuickPlace Customization Triad*. The triad consists of end users, developers, and administrators (see Figure 142 on page 410), who each bring a piece of technology, social, or business science to complete the work practices puzzle.

As with the *line of business* team it supports, the customization triad is often ad hoc, and when effective, adds value to the design of the application beyond the capabilities of any individual user, developer, or administrator.

QuickPlace makes possible a geographically dispersed and more spontaneous implementation of the *participatory design* methodology, whereby users and other stakeholders are involved in the design from a very early stage and throughout the design and development process (Ehrlich 146).

The intent of participatory design is to lower decision thresholds by empowering workers to determine the development of the information system and of their workplace (146). It is consistent with a theory of complex adaptive systems that organizations are more effective operating closer to the more dynamic chaos end of the complexity spectrum than near the more stable side of equilibrium.

This would be true for organizations involved in all but the most tightly coupled and complex activities, like nuclear power plants. But lest you assume we advocate unbridled chaos in the workplace, we would point out that participatory design can be implemented in concert with a more activist approach by management to influence the behavior of teams for the better by seeding content and processes into PlaceType designs.

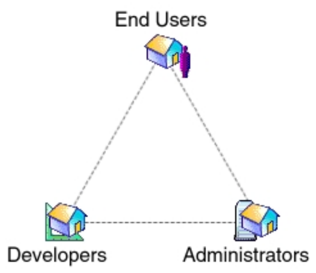


Figure 142. The QuickPlace Customization Triad

Whenever possible, end users achieve their goals by customizing QuickPlace without the intervention of administrators or developers (see later in this appendix). When the requirements for customization exceed the end-users' capabilities, they need not learn a programming language, but instead form a team with those interested in CSCW and with those who have the skills to customize an application to satisfy their goals. At least one of member of the team must facilitate the design process to ensure that the user's goals drive the development, while another should represent the development process, matching specialized skills to meet the design specifications. As Cooper puts it, to allow the same developers to do their own design work introduces a conflict of interest like letting basketball players referee their own basketball game (Cooper 108).

K.1.4.2 Specialization and the need for collaborative development

When the invention of the wheel was first applied to transportation, it was probably the same person who designed and built the wheel, axle, and attached it to whatever crude platform it supported. As successive generations of transportation solution providers refined earlier designs, they found that production required varied skills that a single person did not generally have.A wheelwright made the wheels; a blacksmith created the rims for the wheels and carriage hardware; a coach builder made the passenger compartment and suspension; and a saddler made the harness and bridle work.

As technology improved, so too did the methods of its design and production. The lessons learned by generations of component makers were passed down and emerged as design specifications for component subsystems. Today, an automobile is a system whose component subsystems are assembled at different times from parts made in many different countries of the world by people with highly specialized skills. Without the collaboration of loosely coupled communities, it would be impossible to produce today's automobile.

K.1.4.3 Breakthrough! One developer can leverage QuickPlace

QuickPlace empowers a *single* user with no programming skills and without special software to create, customize, and reuse applications tailored specifically to a team, project, or community. Every QuickPlace instantiation is ready to use immediately upon creation and includes everything you need to manage the content, users, and appearance of the Place. You don't need to know anything about QuickPlace's component subsystems to use it. You just “turn the key” and it runs.

As a developer, when customizing QuickPlace, if you know a little, you can change a lot. If you know even a little about Web development formats such as HTML, JavaScript, and cascading style sheets (CSS), you can immediately make dramatic changes to the way QuickPlace looks. If you know a little bit about development tools used in Domino, such as LotusScript or Java, then right away you can extend or change the way QuickPlace behaves.

If you know something about application integration through Microsoft Visual Basic for Applications (VBA), enterprise connectors to Domino databases, or any techniques that integrate content into the Web, then you have a ready means to bring other applications and their content into QuickPlace. In each case, you would only have to learn the part of the QuickPlace Object Model that directly touches the part you want to customize.

K.1.4.4 QuickPlace leverages the power of many, collaboratively

QuickPlace empowers a *community* of users to develop on the QuickPlace platform, collaboratively. The skills of application developers, like the skills of automobile designers and manufacturers, are increasingly specialized and complex in their interactions. Ask several developers with varied skill sets how they would solve a particular challenge, and you are likely to get several answers. Each developer is likely to propose a solution based on the particular technology in which that developer is strongest.

When you have access to only a limited variety of development skills, you are less likely to create an effective solution. You need a way to bring diverse skills and skill levels to bear, even when they are separated by time and distance. Developers, too, need a virtual place in which to collaborate.

QuickPlace developers and users can form communities dedicated to sharing examples of their customization of QuickPlace, or they can collaborate concurrently with the development of a Place that serves another community. For example, the QuickPlace development team at Iris, uses a Place, called Haiku Team, to coordinate their activities associated with the development of the QuickPlace source code. The design of Haiku Team is itself a functioning prototype that includes many thematic and programmatic customizations. You can visit a QuickPlace community that serves all QuickPlace developers at:

`http://www.quickplace.com/devzone`

Now let's imagine you are using QuickPlace from your New York office and want to change the way QuickPlace handles a special form. You know that a PlaceBot is a way of adding automation to QuickPlace, using LotusScript or Java, but you don't have the necessary programming skills.

You are able to locate a Java Developer in your New Zealand office, who will write an applet just for you. You can invite the Java developer to become a member in your Place with access to create the PlaceBot. You create a room (or even a separate Place) to work on development issues, or where administrators and end users can test the PlaceBots as they are developed.

In this way you can rapidly form teams within a Place, comprised of end users, developers, and administrators who meet at a Place, collaborate, and then disband as issues are resolved.

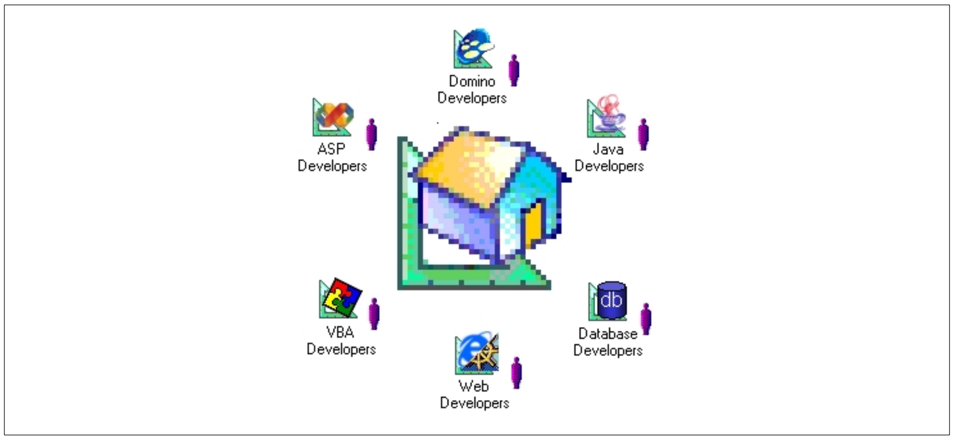


Figure 143. Developers with diverse skills using QuickPlace collaboratively to customize it

K.1.4.5 The ecology of QuickPlace development

As the users and developers of your Places drive innovation from the bottom up, you can disseminate best practices from the top down. You can use QuickPlace PlaceTypes as the building block for an ecology of applications that are always evolving independently, while increasing the fitness of the whole system.

An ecology of QuickPlace applications exhibits the properties of complex adaptive systems (CAS); complex large-scale behaviors emerge from the aggregate interactions of its less complex parts (aggregation), and a multiplier effect resulting from hidden interactions means that the value of the whole is more than the value of its parts (nonlinearity). The members of a team, who are members of a Place where that team collaborates, are also members of other teams and their Places. No Place exists in a vacuum, and connecting one Place to another can be as easy as sending mail.

You don't have to have a specific repository to store your QuickPlace designs, because they are embedded in every Place that you create with QuickPlace. Any manager of a Place can make the Place’s design available to other creators of Places as a PlaceType.

PlaceTypes are templates that can repeatedly deliver a bundle of tools and content, designed for a particular type of team, process, or situation. Like the DNA in organisms that allows them to evolve, PlaceTypes codify the evolution of the applied understanding of work practices. When something works, whether by design or by chance, the organization may want to designate it as a best practice and make it available for others in the organization to use.

Teams can use rapid prototyping or continuous tweaking to innovate, and in customizing QuickPlace, they can create PlaceTypes specialized according to their new understanding of processes and practices. If you can reach a critical mass of users who customize QuickPlace, this interaction and evolution can occur naturally.

If all of this change seems a bit out of control to you, then you're exactly right. The optimal environment for innovation relies on teams capable of semi-autonomous action. Complex adaptive systems adapt to meet the demands of their environments without central direction, exhibiting a behavior described as self-organizing. Binding applications to rigid practices wrongly assumes that all causes and effects are traceable and ignores the reality that local work practices contain many variations that we can only begin to capture. Once again, the users' personal, practical, and corporate goals must drive the implementation, and not the other way around.

While technological improvements are primarily linear, software development is subject to the full range of nonlinear behaviors (Czerwinski 30), precisely because it depends on interactions with humans and groups of people as complex systems. *Linearity* means something like the whole is equal to the sum of its parts. You can pick up a Swiss Army Knife and intuitively understand exactly what it is intended to do, and a good part of how to use it, just by manipulating it. Additional technological improvements to this mechanical age device produce predictable results.

However, a television remote control, a digital-age device driven by software, is a different story. It's easy enough for your thumb to find the buttons and intuitively discover that you can push the buttons to make something happen. However, it's nearly impossible to predict the results of pushing an unfamiliar button. Have you ever mistakenly turned on the closed captioning subtitles or changed the menu to a foreign language and then had to study the owners' manual or ask a friend to change it back?

If you think the remote control is hard for people to figure out, consider the Web, where the purpose of a typical blue link is a pure meta-function that is totally hidden to the user until it is clicked. Where software and software-driven devices meet the human mind, a nonlinear resistance makes it impossible to predict the behavior of the whole from the sum of its parts. Cooper calls it “cognitive friction” (19). Likewise, an earlier thinker, Clausewitz, used “friction” and “fog of war”, borrowing the language of the physical sciences to describe complex behaviors that are driven by human interaction. The complexity of these interactions only underscores the previously discussed need for interaction design on one hand, and on the other, the allowance for lowered decision thresholds to customize. Fortunately, QuickPlace can easily support both.

You can't control evolution in the QuickPlace ecology, but you can influence it. Both self-organizing and hierarchical systems need to be deployed in order to restrain the worst excesses of each other. Merging lanes of automobile traffic usually flow undirected and without accidents, but the adherence to posted speed limits helps to keep people from getting killed. The central question of what you can influence by deploying QuickPlace, or any groupware platform, depends on the extent to which your line of business organization and developer organization is prepared for deployment.

With QuickPlace, installation, administration, and development are easy and are not the most significant barriers to deployment. The toughest obstacles most organizations will have to surmount are cultural. An organization in which information sharing is not already well-established and rewarded will have a hard time getting any benefit out of QuickPlace. As IBM Knowledge Management Consultant David Snowden points out:

“In general, if a community is not physically, temporally and spiritually rooted, then it is alienated from its environment and will focus on survival rather than creativity and collaboration. In such conditions, knowledge hoarding will predominate and the community will close itself to the external world. If the alienation becomes extreme, the community may even turn in on itself, atomising into an incoherent babble of competing self interests.” (Snowden 1999)

In many ways, the utility of a groupware application platform like QuickPlace depends on how it is customized. Content gives QuickPlace its shape; work practices and process drive its behavior. If deployed without customization to accommodate the subtle social protocols of the organization, people will reject it, finding some other way to do work, or continuing to use an established means. If deployed in an organization with dysfunctional teams, then QuickPlace will mirror those teams. If your deployment does not have the support of the senior leadership, it may never reach the critical mass necessary to get value from applications like shared calendars.

Unlike most groupware, QuickPlace really works out-of-the-box, but getting up and running is not in itself an outcome supportive of your mission, vision, and purpose. Neither knowledge management, nor even collaboration, can come out of a box.

Summary of introduction

We have discussed the implications of customizing QuickPlace as a platform for content management and distributed application development and how they are deeper than customizing one instantiation of a collaborative workspace. Customizing applications built on the QuickPlace platform can bring profound changes in the way teams conduct business, but only if we appreciate their attendant complexity. QuickPlace brings instant collaboration to the work of teams, including the teams who customize QuickPlace.

Your success in customizing QuickPlace depends on an understanding of your organization's people, processes, and strategy. You must understand who the customers are, what outcome they expect, how you will know when you get there, and what the consequences of failure are. If you can customize QuickPlace to facilitate a group's decisions on mission, purpose, scope and communications practices, then your influence will be positive and far-reaching.

K.2 Why customize QuickPlace

"A dollar spent on the same system may give a competitive advantage to one company but only expensive paperweights to another." Erik Brynjolfsson

QuickPlace is designed for maximum flexibility as a collaborative workspace and as a development platform. However, you probably already have some idea of the potential for customizing:

- **Extend** QuickPlace's already robust collaboration, document management, workflow, and administration capabilities
- **Integrate** QuickPlace with existing platforms
- **Reuse** components developed in or for QuickPlace and even reuse Places themselves with PlaceTypes

K.2.1 What drives the need to extend, integrate, and reuse

Motivations for customizing QuickPlace vary as widely as the contexts in which it is deployed and the outcomes that organizations seek. You may just want to tweak a particular feature such as Themes, or add specific functionality to match the subtle social protocols of your organization's communities.

Alternately, you may want to extend the collaboration, document management, and workflow built into QuickPlace. Then, at some point, you might want to stop tweaking and start integrating QuickPlace into the document management, workflow, and databases found in your organization. Finally, when you find something that works, you will want a way to reuse it.

K.2.1.1 Why not just run QuickPlace off-the-shelf

As saying goes, “If it ain’t broke. don’t fix it.” Sometimes it just makes sense to deploy a solution as it comes out of the box. However, across different organizations, and even within the same organization, there are sufficient differences in work, culture and context that require some degree of customization to make using QuickPlace attractive enough for a critical mass of users to adopt it. Just getting QuickPlace to work may not be enough to make QuickPlace work for you and your organization.

When Lotus released QuickPlace 1.0, many customers wanted directory integration with a Domino directory. In most organizations interested in using QuickPlace within the context of another intranet or extranet, failure to work seamlessly with existing user names and passwords would be a deal breaker.

Fortunately, QuickPlace 2.0 provides directory integration, as well as other key integration features, but some early adopters had to work out solutions to solve many local requirements before they were satisfied by a new product release. Your requirement may be unique enough that you will have to find a solution yourself. No doubt you can think of local requirements that all of your applications must support, such as marking every document with predefined subject headings or security classifications. The good news is that, a wide range of customization options for end users and developers allow you to customize QuickPlace to make it work for you.

K.2.1.2 Tweak to add specific functionality

With QuickPlace, you can create applications better than off-the-shelf programs costing thousands of dollars for two reasons. The first reason is the power of QuickPlace/Domino plumbing, something that's always there in every Place, making even simple applications powerful. Tracking versions, security, Web, sharing, permissions, automatic knowledge base, replication, and remote functionality all are there, virtually automatically.

The second reason we feel QuickPlace is better is subtler. No programmer knows what your end user really needs, even if a user attempts to tell the programmer. And few end users have the programming skills of a professional programmer. However, QuickPlace lets end users easily construct relatively complex, mission-critical applications at a level of power no programming novice would be able to do with another technology.

Probably, no one is more insistent on doing things “their way” than an individual end user. The computer revolution destroyed a lot of that individuality by forcing end users to change a large part of the way they operated to fit the programs forced upon them. QuickPlace gives power back to the individual. QuickPlace applications do what you want, the way you want.

K.2.2 People want to extend QuickPlace

When you extend QuickPlace, you are concerned with how an application *looks* and how it *behaves*. In both cases, your goal is either to refine the existing capability to conform to your local requirements or create additional capability. To manipulate the look of a Place, end users can select from a wide range of themes and can tweak individual components to achieve highly personalized appearance for their places. However, if you wanted to create a layout unlike any existing theme or use advanced presentation, such as DHTML, you will need to customize QuickPlace's user interface, using the QuickPlace Layout Architecture.

You may want to tweak QuickPlace to conform to the communications norms of the group who uses it, either in a particular instantiation or in all instantiations. You might want to rename the default *Library* folder to something the people who will use it will find more significant, like *Reading Room*, *Read-Ahead Material*, or *Before You Arrive.* In some organizations, the word security is never used to refer to access control, but instead is a legally defined term of art for a level of sensitivity. Such an organization would want to change references to *security* in QuickPlace to *access control*, while adding a *security classification* field on all forms.

While end users have many options to customize Places to fit their specific needs for document versioning, notification, and workflow, you may want to go further. You may want to extend QuickPlace with behaviors such as automatic archiving of documents to reduce information overload when they reach a certain threshold. This threshold varies greatly from one organization to another and is applied subjectively from one individual to the next.

PlaceBots and custom forms are a ready means to extend the already rich document handling options in QuickPlace, without having to access the QuickPlace object model from the back end (see Chapter 3, "Installing the QuickPlace server" on page 21). However, for even deeper customization, you can use C++ to modify the QuickPlace Event Interface (See Chapter 9, "QuickPlace Developer's Kit and the event interface" on page 265).

K.2.3 People want to integrate QuickPlace

If your desire to extend the capabilities of QuickPlace extends farther than you can support with custom development, or you already have another solution in place, you may turn to integrating QuickPlace with other platforms.

K.2.3.1 Integrate with desktop applications

QuickPlace readily provides extensive integration with Microsoft Office 97 and 2000. Documents and forms created in Word, PowerPoint, Excel, and Front Page can be used in QuickPlace without modification. With knowledge of technologies such as Visual Basic for Applications (VBA) or COM, you can go much farther in bringing desktop applications into QuickPlace and QuickPlace into desktop applications.

K.2.3.2 Encapsulation: Wrap and be wrapped

The simplest means of integrating any Web application with another is by creating hyperlinks. Though it's simple to create links from QuickPlace to other Web applications, the perceived effects can be powerful, as you wrap QuickPlace within the context of a Web portal, or you modify QuickPlace to be the wrapper of other Web content.

QuickPlace can be encapsulated by another platform, such as Domino Workflow, which could trigger the instantiation, use, and closeout of a QuickPlace within the context of a workflow job.

K.2.3.3 Enterprise integration

Large organizations invest heavily in enterprise platforms for storing data, processing transactions, supporting work processes, and managing documents. You can integrate QuickPlace seamlessly into many platforms to take advantage of their structure and power, while complementing them with QuickPlace's flexibility and affinity for unstructured work practices. You can connect directly to the structured data of popular database platforms by using simple LotusScript, Java PlaceBots and so on.

K.2.4 People want to reuse QuickPlace—component evolution

Creating applications with QuickPlace is easy, but creating applications that support processes and work practices that are constantly evolving requires discipline and an effective means of leveraging the lessons of the past to create the next solution.

QuickPlace is designed for reuse at every level, from the custom, forms, layout, and PlaceBots uploaded to extend the functionality of individual Place, to the Themes and PlaceTypes that are defined as the templates for future instantiations. Everything you do to customize QuickPlace can be reused and recombined to create even better solutions. And just as the Web is an open, platform-independent forum for worldwide collaboration, QuickPlace is an open environment for developers to leverage heterogeneous technology and skills.

K.3 Four ways you can customize QuickPlace

QuickPlace becomes customized for a particular purpose as soon as users add content to it. Users can set the external and internal parameters in a place to adapt of QuickPlace's appearance and behavior to fit the work practices of the team. Managers, administrators, and developers can reuse customizations through PlaceTypes. Developers can customize QuickPlace more deeply to dramatically rewire its behavior or integrate it with other applications. Here we discuss four kinds of groupware customization as they are applied to QuickPlace:

- Content-based customization
- Setting external parameters
- Setting internal parameters
- Totally customized solutions

K.3.1 Members and managers customize with content

Depending on what content is already bundled in its PlaceType, a newly instantiated place is merely a shell and doesn't really become useful until people begin to fill it with content. Content-based customization requires no special technical skills and is done by end users. However, everyone who is involved with customization must appreciate the critical role content plays in shaping the interaction of users with in the place. As developers, you can create fully customized applications for the purpose of facilitating the content management.

K.3.2 Members and managers set external parameters

Some customization is accomplished by selecting options and adjusting settings at the time of content creation. You might consider these external parameters to be just *using* QuickPlace, and not about customization at all. However, when considered on the whole, the external parameters offer users the ability to customize the application to suit the way they intend to use it. This is still end-user customization, but it may require leadership to guide team members as to the practices and norms of the group.

Examples of customization by setting external parameters in QuickPlace include:

- *Publish as* options
  - Include author and date published banner on document
  - Notify (via E-mail)
  - Add editors
  - Restrict readers
  - Add to calendar
  - Save as draft
- Select a form for new document

K.3.3 Managers, administrators, and developers set internal parameters

QuickPlace allows managers to set up the ground rules for communications in their Places. For example, you might have a PlaceType developed for the purpose of supporting conferences with a custom form for submitting presentation papers. The level of customization necessary to support the handling of submissions might range from simply inputting a list of reviewers in **Customize->Forms->[form name]->Workflow Modify->Multiple Editors** to using an **Approval Cycle** to route the form through a series of members in a specific order before placing the form in a specific folder. It is possible to customize an entire range of workflow options by setting the internal parameters in QuickPlace. Examples of customizing internal parameters:

- Defining the mission, vision and scope of the PlaceType or instantiated place through a *Welcome* page.
- Substituting the **Tutorial** link with a *preferred process* to disseminate the formal processes, work practices, and communications norms of the team.
- Creating folders and Customizing folder options:
  - Change folder title to imply what goes in and how it is used
  - Designate a “favored” form or a special response form
- Customizing form handling, using:
  - Workflow options under **Customize**
  - Simple PlaceBots, provided by developers
- Selecting themes and skins for:
  - Customized look and feel
  - Corporate branding
  - Selecting server settings in the Administration Place

K.3.4 Developers and administrators create fully customized solutions

A fully customized solution built on QuickPlace is one that requires system administration beyond setting internal parameters and development expertise beyond uploading shared PlaceBots or Themes. When a fully customized application is intended to support users' personal, practical, and corporate goals that are markedly different from previous applications, then the development process should be preceded by an interactive design process to determine what the users' goals are and what the application would need to do to satisfy those goals.

K.4 Deploying QuickPlaces with a Turnkey server

"The device is delivered—instantaneously causing what we call the Genesis effect. Matter is reorganized with life-generating results. Instead of a dead moon, [you have] a living breathing planet, capable of sustaining whatever life forms we see fit to deposit on it."—Describing the Genesis device, a turnkey planetary life generator in Star Trek II: The Wrath of Khan

K.4.1 Overview: Deploying your customized QuickPlace

Chapter 11, "Creating PlaceTypes and a Turnkey Server" on page 321 is all about packaging your customized QuickPlace solutions into a self-deploying, portable, cohesive system, which we call a Turnkey server. A Turnkey Server installs ready-to-use, bundled with PlaceTypes customized to support selected communities, teams, events, and processes. It may also include additional application layers to enhance the enterprise integration and administration of QuickPlace. You can use everything you know about customizing QuickPlace to creating a Turnkey Server that meets the needs of a particular customer organization or community.

In the remainder of this appendix we will discuss what makes a Turnkey Server more than just an installation disk with your company's logo on it.

K.4.2 QuickPlace Turnkey Server deployment—The Genesis effect

Does life evolve along a predestined path, or does it suddenly emerge from what appeared lifeless and programmatic?

Remember Project Genesis from the film, *Star Trek II: The Wrath of Khan*? If you're a techie, there's a good chance you're also a *Trekkie.* In the film, which is set in the future, scientists had figured out how to rearrange the molecular structure of inert matter into life-generating matter. In the final phase of their experiments, they would launch the Genesis device, which would deploy on the surface of a lifeless planet or moon, setting in motion the unfolding of life in an accelerated evolution to a mature planetary ecosystem, teeming with life forms of all shapes and sizes. *As the story goes, "...put simply, Genesis is life from lifelessness..."* and that is exactly what a TurnkeyServer is about.

In the film, the planet that emerges after the Genesis device is deployed bears a strong resemblance to our own planet, presumably by design. Initially, the only seeds to bloom in the QuickPlace environment that unfolds from a Turnkey Server will be the seeds you put in it. You are in complete control of the initial set of genes that will shape the appearance and behavior of the first generation of Places to inhabit the environment. You could expect more complex interaction in an environment with a diverse population of PlaceTypes, each customized for a particular purpose, than you could if you included only the Standard QuickPlace for Teams, that ships with QuickPlace.

Previously, we discussed QuickPlace as a complex ecology, where applications evolve, more or less independently, while exchanging their digital DNA through the definition of PlaceTypes. Creating and deploying Turnkey Servers presents some interesting questions that amount to the same thing as a debate between a Calvinist, arguing that all outcomes are predestined, and a Deist, saying the Divine Watchmaker built the world, wound it up, and has let it run on its own. We know that once human users make contact with the QuickPlace world, it will never be the same. As users and developers customize QuickPlace by adding content, setting parameters, and developing new, fully customized solutions, the world takes on a life of its own. A critical question for the designer of the Turnkey Server is--what kind of Places to include and what processes and practices need to be wired into their design?

K.4.3 What customers want from QuickPlace

We cannot begin to fathom what goes into a Turnkey Server until we understand who would want one and why. For the purposes of this discussion, we will presume that someone wants something and that you can give it to them. We will call that person your customer, but understand that the customer can be any person or persons internal or external to your organization to whom you provide a product or service.

You could be a development or consulting organization, delivering a standalone QuickPlace solution or a QuickPlace solution embedded within consulting services or in which content such as training is embedded. Alternatively, you could be a user organization trying to replicate your QuickPlace application in a sister organization or perhaps in your company's newest office halfway across the world.

In either case, you need to know who the customer is and what outcome they expect from the deployment of a turnkey server. The first step in that direction is to determine what the key process areas of the organization are?

Ultimately, customers are more concerned with outcomes than the means to outcomes. Although outcomes are what shape their desires in the long term, they are often distracted by inputs and outputs in the short term. Sadly, customers are often more preoccupied with superficiality than whether or not a solution compliments or enhances its value-driving key process areas. As a solution provider, you will see your solution cast onto the rising scrap heap of failed solutions if you do not identify these key process areas and support them.

A customer might point to a working ecology of applications, deployed in an organization where improved collaboration has increased innovation and improved effectiveness. The customer will tell you that's what they want. However, they often can't distinguish between the computer applications and the human activities that accompany them. As previously mentioned, if you took the best corporate portals that have been deployed, froze them, and transplanted them to other organizations, they wouldn't take root. For the same reason that technology cannot create a team from people who have no shared interest or relationship, the portal is more than applications and content, it needs people to participate to make it live.

K.4.3.1 How to prime the pump

Even if you could take all of the content of the thriving portal and port it with the seedling, the users wouldn't necessarily understand what to do to renew the content in the context of their own work. That's where you come in.

Like the default Welcome page in QuickPlace, which does not assume that users even know why they would want a shared virtual meeting place, each PlaceType must be embedded with the guidance for how to make the Place (and hence the key process area) work, and reflect the process or practices in its design.

To do this, you could create an application that would assume the approach of a virtual interview with the new team leader and Place manager. During this interview, you could present information designed to educate the leader about how to conduct business and how to be a good facilitator, while helping the leader craft a mission and purpose for the team. Your application would then inject that information into the Welcome page of the newly instantiated place.

Appendix L. Additional Web material

Additional Web material is referenced in this redbook and can be found on the IBM Redbooks Web site. Table 51 lists the file names and describes the content.

Table 51. Additional Web material

File name	Description
6000-all.zip	This file contains all the zip-files listed below **except 6000repo.zip, 6000ddoc.zip and 6000tkey.zip**. If you want to look at most of the examples, you can save a bit of time by downloading this file.
6000cssp.zip	Source files for style sheet previewer.
6000mthm.zip	The files used to create the Millennia Theme used in this book. Relates in particular to Chapter 4, “Creating Themes” on page 47.
6000favo.zip	Source files for Favorites page described in 4.4, “Adding the Favorites HTML page” on page 104.
6000gwiz.zip	Files for Graphics Wizz utility used to build URLs for dynamically created graphics as described in 5.1.3, “Graphics Server Image Wizard: GraphicsWhizz” on page 132.
6000poll.zip	HTML forms: using the QuickPlace upload control, using the QuickPlace Rich Text control and finally the Poll form as described in 6.4, “Upload a manually created HTML Form” on page 146.
6000mapr.zip	Source files for the Mapperizer PlaceBot as described in 7.4, “A site map PlaceBot” on page 171.
6000mlat.zip	Source files for the mail room attendant PlaceBot as described in 7.5, “The QuickPlace Mail Room Attendant” on page 186.
6000repo.zip	Source files for TheXRay reporting application located in a Domino database as described in 7.6.1, “TheXRay server status reporter agent” on page 195.
6000msov.zip	Files that illustrate how to do mouse-over effects in QuickPlace as described in 8.2.20, “Building URLs: Referencing images” on page 249.

File name	Description
6000qdll.zip	Sample QuickPlace server add-in programs to modify mail messages including source as described in Chapter 9, “QuickPlace Developer’s Kit and the event interface” on page 265.
6000rqdb.zip	Domino database with example of how handle QuickPlace creation requests as described in 10.2, “A Domino application to handle requests for new Places” on page 281.
6000csyn.zip	Contains source for Java agent the synchronizes a Domino calendar with a QuickPlace calendar as described in 10.1, “Mirroring data to Notes - a Java agent example” on page 277.
6000ddoc.zip	Files from the example showing QuickPlace integration with Domino.Doc as described in 10.3, “Integrating with Domino.Doc” on page 287.
6000tkey.zip	All the files we added to build the Millennia Turnkey server as described in 11.6, “Deploying your customized QuickPlace as a Turnkey Server” on page 340. Copy your QuickPlace installation CD to your hard disk. Unpack this zip file into the installation directory, keeping the directory structure, and you will be ready to test the Turnkey server.

Note: The files are compressed in zip package files and must be unpackaged using a tool like WinZip or similar.

L.1 How to get the Web material

The Web material associated with this redbook is also available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG246000`

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number.

Appendix M. Special notices

attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

e (logo)®	Redbooks Logo
IBM ®	AIX
AS/400	CICS
DB2	DB2 Universal Database
MQSeries	S/390
SecureWay	VisualAge
WebSphere	Redbooks

The following terms are trademarks of Lotus Development Corporation in the United States and/or other countries:

Lotus ®	Lotus Notes ®
Lotus Domino	LotusScript ®
Domino Workflow	Domino.Doc
Lotus Smartsuite ®	Lotus QuickPlace
People Places and Things ®	Lotus Sametime
SUPER.HUMAN.SOFTWARE	

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

Appendix N. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

N.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 435.

- *Lotus Domino Release 5.0: A Developer's Handbook*, SG24-5331, Lotus part number CC7EDNA
- *XML Powered by Domino - How to use XML with Lotus Domino*, SG24-6207
- *Using VisualAge for Java to Develop Domino Applications*, SG24-5424, Lotus part number CT6ENNA
- *COM Together - with Domino,* SG24-5670
- *LotusScript for Visual Basic Programmers,* SG24-4856
- *Developing Web Applications Using Lotus Notes Designer for Domino 4.6*, SG24-2183
- *Using Domino Workflow*, SG24-5963, Lotus part number CT6GNML
- *Creating Customized Solutions with Domino.Doc,* SG24-5658
- *Connecting Domino to the Enterprise Using Java*, SG24-5425, Lotus part number CT6EMNA
- *Lotus Domino R5.0 Enterprise Integration: Architecture and Products*, SG24-5593, Lotus part number CT6QUNA
- *Performance Considerations for Domino Applications*, SG24-5602, Lotus part number CT7V6NA
- *Lotus Notes and Domino R5.0 Security Infrastructure Revealed*, SG24-5341, Lotus part number CT6TPNA
- *Domino and WebSphere Together,* SG24-5955
- *Lotus Sametime Application Development Guide*, SG24-5651, Lotus part number CT7AKNA
- *Getting the Most From Your Domino Directory*, SG24-5986

N.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at `ibm.com/redbooks` for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	**Collection Kit Number**
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

N.3 Other resources

These publications are also relevant as further information sources:

- *The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Stop the Insanity*, Cooper, A., 1999, New York: SAMS, a division of MacMillan Computer Publishing, ISBN 0-67231-649-8
- *Coping With the Bounds: Speculations on Nonlinearity in Military Affairs* by Tom Czerwinski, 1998, available online at: `http://www.dodccrp.org/copind.htm`
- *Designing Groupware Applications: A Work-Centered Approach* by Ehrlich, K. (2000) in D. Smith (Ed.) *Knowledge, Groupware, and the Internet*, Boston: Butterworth-Heinemann, ISBN 0-75067-111-4
- *Designing Web Usability* by Jakob Nielsen, New Riders Publishing, Indianapolis USA, December 1999, ISBN 1-56205-810-X

N.4 Relevant Web sites

These Web sites are also relevant as further information sources:

- `http://www.quickplace.com/devzone` DevZone for administrators and developers, brought to you by the QuickPlace Development Team. At the DevZone. you'll find technical news, tools, and documentation.
- `http://www.lotus.com/developer/` Lotus' primary destination for the latest developer information and resources. Contains articles about new and

current technologies, along with relevant tips and techniques to help you build dynamic collaborative e-business applications.

- `http://notes.net/` Notes.net from Iris - the developers of Notes and Domino - is a technical Web site with discussion forums, documentation and the Webzine Iris Today with many good articles about technical details of Domino.
- `http://ibm.com/developer/` The IBM developerWorks Web site is designed for software developers, and features links to a host of developer tools, resources, and programs.
- `http://support.lotus.com/` Lotus Support's Web site - Search using keywords or browse the Lotus Knowledge Base and locate helpful and informative tech notes and technical papers for the entire Lotus Product family. This source of information contains the latest technical information updated hourly.
- `http://www.webmonkey.com/` Web developer resource site with many good how-to articles about topics like Authoring, Design, Multimedia, E-Business, Programming and Backend connectivity.
- `http://www.webreview.com/` Weekly Webzine dedicated to Web professionals. Includes guides in areas like Style sheets, Web Browsers, Web tools and Ranking systems.

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** **ibm.com**/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the "How to Order" section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl`

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at `http://w3.ibm.com/` for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	**Order Number**	**Quantity**

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-6000-00 Customizing QuickPlace
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	 O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

IBM

Redbooks

Customizing QuickPlace

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Customizing QuickPlace

Create your unique look and feel with Themes and forms

Use PlaceBots to add application logic

Capture best practices using PlaceTypes

Lotus QuickPlace is the leading self-service Web tool for team collaboration. This IBM Redbook is about how you as a Place owner, a Web designer, a programmer or a QuickPlace administrator can take QuickPlace to the next level through customization.

We show you how to apply your own unique graphic design to the QuickPlace user interface as well as functionality using JavaScript. We show how you can develop forms (built-in, MS Office or HTML forms) and use the built-in workflow to support your process. Then we show how you can add automation by programming PlaceBots (agents) in LotusScript or Java. To develop advanced customizations in QuickPlace you have to understand the internals and how to modify them. We discuss this on chapters about the QuickPlace Object Model and the QuickPlace Developer's Kit.

Even though QuickPlace is a selfservice tool, there will often be a need to integrate with other systems and we show how you can integrate with Notes, Domino, Domino.Doc document management, the Domino Workflow product and we also discuss how to integrate QuickPlace in a portal.

Finally, we show how you can clone the Places you choose, together with their customization as PlaceTypes, and how to build a Turnkey server with your PlaceTypes and other customizations, for internal reuse or for resale.

Many examples are included throughout the book.

SG24-6000-00 ISBN 0738421049